DonorsChoose

DonorsChoose.org receives hundreds of thousands of project proposals each year for classroom projects in need of funding. Right now, a large number of volunteers is needed to manually screen each submission before it's approved to be posted on the DonorsChoose.org website.

Next year, DonorsChoose.org expects to receive close to 500,000 project proposals. As a result, there are three main problems they need to solve:

- How to scale current manual processes and resources to screen 500,000 projects so that they can be posted as quickly and as efficiently as possible
- How to increase the consistency of project vetting across different volunteers to improve the experience for teachers
- How to focus volunteer time on the applications that need the most assistance

The goal of the competition is to predict whether or not a DonorsChoose.org project proposal submitted by a teacher will be approved, using the text of project descriptions as well as additional metadata about the project, teacher, and school. DonorsChoose.org can then use this information to identify projects most likely to need further review before approval.

About the DonorsChoose Data Set

The `train.csv` data set provided by DonorsChoose contains the following features:

Feature	Description
`project_id`	A unique identifier for the proposed project. **Example:** p
`project_title`	Title of the project. **Examples:** • `Art Will Make You Happy!` • `First Grade Fun`
`project_grade_category`	Grade level of students for which the project is targeted. enumerated values: • `Grades PreK-2` • `Grades 3-5` • `Grades 6-8` • `Grades 9-12`

Feature	Description
`project_subject_categories`	One or more (comma-separated) subject categories for following enumerated list of values: • `Applied Learning` • `Care & Hunger` • `Health & Sports` • `History & Civics` • `Literacy & Language` • `Math & Science` • `Music & The Arts` • `Special Needs` • `Warmth` **Examples:** • `Music & The Arts` • `Literacy & Language, Math & Science`
`school_state`	State where school is located (Two-letter U.S. postal co (https://en.wikipedia.org/wiki/List_of_U.S._state_abbrevi **Example:** `WY`
`project_subject_subcategories`	One or more (comma-separated) subject subcategories **Examples:** • `Literacy` • `Literature & Writing, Social Sciences`
`project_resource_summary`	An explanation of the resources needed for the project. • `My students need hands on literacy materi sensory needs!`
`project_essay_1`	First application essay*
`project_essay_2`	Second application essay*
`project_essay_3`	Third application essay*
`project_essay_4`	Fourth application essay*
`project_submitted_datetime`	Datetime when project application was submitted. **Exam** `12:43:56.245`
`teacher_id`	A unique identifier for the teacher of the proposed projec `bdf8baa8fedef6bfeec7ae4ff1c15c56`
`teacher_prefix`	Teacher's title. One of the following enumerated values: • `nan` • `Dr.` • `Mr.` • `Mrs.` • `Ms.` • `Teacher.`

Feature	Description
`teacher_number_of_previously_posted_projects`	Number of project applications previously submitted by t **Example:** `2`

[*] See the section **Notes on the Essay Data** for more details about these features.

Additionally, the `resources.csv` data set provides more data about the resources required for each project. Each line in this file represents a resource required by a project:

Feature	Description
`id`	A `project_id` value from the `train.csv` file. **Example:** `p036502`
`description`	Desciption of the resource. **Example:** `Tenor Saxophone Reeds, Box of 25`
`quantity`	Quantity of the resource required. **Example:** `3`
`price`	Price of the resource required. **Example:** `9.95`

Note: Many projects require multiple resources. The `id` value corresponds to a `project_id` in train.csv, so you use it as a key to retrieve all resources needed for a project:

The data set contains the following label (the value you will attempt to predict):

Label	Description
`project_is_approved`	A binary flag indicating whether DonorsChoose approved the project. A value of `0` indicates the project was not approved, and a value of `1` indicates the project was approved.

Notes on the Essay Data

Prior to May 17, 2016, the prompts for the essays were as follows:

- **project_essay_1:** "Introduce us to your classroom"
- **project_essay_2:** "Tell us more about your students"
- **project_essay_3:** "Describe how your students will use the materials you're requesting"
- **project_essay_3:** "Close by sharing why your project will make a difference"

Starting on May 17, 2016, the number of essays was reduced from 4 to 2, and the prompts for the first 2 essays were changed to the following:

- **project_essay_1:** "Describe your students: What makes your students special? Specific details about their background, your neighborhood, and your school are all helpful."
- **project_essay_2:** "About your project: How will these materials make a difference in your students' learning and improve their school lives?"

For all projects with project_submitted_datetime of 2016-05-17 and later, the values of project_essay_3 and project_essay_4 will be NaN.

In [0]:

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")

import sqlite3
import pandas as pd
import numpy as np
import nltk
import string
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import confusion_matrix
from sklearn import metrics
from sklearn.metrics import roc_curve, auc
from nltk.stem.porter import PorterStemmer

import re
# Tutorial about Python regular expressions: https://pymotw.com/2/re/
import string
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.stem.wordnet import WordNetLemmatizer

from gensim.models import Word2Vec
from gensim.models import KeyedVectors
import pickle

from tqdm import tqdm
import os

from plotly import plotly
import plotly.offline as offline
import plotly.graph_objs as go
offline.init_notebook_mode()
from collections import Counter
```

```
Output hidden; open in https://colab.research.google.com (https://colab.rese
arch.google.com) to view.
```

In [0]:

```
import pickle
# Load the Drive helper and mount
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, cal
l drive.mount("/content/drive", force_remount=True).
```

In [0]:

```
dir_path = '/content/drive/My Drive/appliedai/Data'
!ls /content/drive/My\ Drive/appliedai
```

```
Data  Dumps  KNN  Logistic_Regression  Naive_Bayes  NB_dumps  New_dumps  tsn
e
```

1.1 Reading Data

In [0]:

```
project_data = pd.read_csv(os.path.join(dir_path,'train_data.csv'))
resource_data = pd.read_csv(os.path.join(dir_path,'resources.csv'))
print(project_data.shape)
print(resource_data.shape)
```

```
(109248, 17)
(1541272, 4)
```

In [0]:

```
print(len(project_data[project_data['project_is_approved'] == 0]))
print(len(project_data[project_data['project_is_approved'] == 1]))
```

```
16542
92706
```

In [0]:

```
print("Number of data points in train data", project_data.shape)
print('-'*50)
print("The attributes of data :", project_data.columns.values)
```

```
Number of data points in train data (109248, 17)
--------------------------------------------------
The attributes of data : ['Unnamed: 0' 'id' 'teacher_id' 'teacher_prefix' 's
chool_state'
 'project_submitted_datetime' 'project_grade_category'
 'project_subject_categories' 'project_subject_subcategories'
 'project_title' 'project_essay_1' 'project_essay_2' 'project_essay_3'
 'project_essay_4' 'project_resource_summary'
 'teacher_number_of_previously_posted_projects' 'project_is_approved']
```

In [0]:

```
print("Number of data points in train data", resource_data.shape)
print(resource_data.columns.values)
resource_data.head(2)
```

```
Number of data points in train data (1541272, 4)
['id' 'description' 'quantity' 'price']
```

Out[19]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

1.2 preprocessing of `project_subject_categories`

In [0]:

```
catogories = list(project_data['project_subject_categories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python
cat_list = []
for i in catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", "
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp+=j.strip()+" " #" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_') # we are replacing the & value into
    cat_list.append(temp.strip())

project_data['clean_categories'] = cat_list
project_data.drop(['project_subject_categories'], axis=1, inplace=True)

from collections import Counter
my_counter = Counter()
for word in project_data['clean_categories'].values:
    my_counter.update(word.split())

cat_dict = dict(my_counter)
sorted_cat_dict = dict(sorted(cat_dict.items(), key=lambda kv: kv[1]))
```

1.3 preprocessing of `project_subject_subcategories`

In [0]:

```
sub_catogories = list(project_data['project_subject_subcategories'].values)
# remove special characters from list of strings python: https://stackoverflow.com/a/473019

# https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
# https://stackoverflow.com/questions/23669024/how-to-strip-a-specific-word-from-a-string
# https://stackoverflow.com/questions/8270092/remove-all-whitespace-in-a-string-in-python

sub_cat_list = []
for i in sub_catogories:
    temp = ""
    # consider we have text like this "Math & Science, Warmth, Care & Hunger"
    for j in i.split(','): # it will split it in three parts ["Math & Science", "Warmth", '
        if 'The' in j.split(): # this will split each of the catogory based on space "Math
            j=j.replace('The','') # if we have the words "The" we are going to replace it w
        j = j.replace(' ','') # we are placeing all the ' '(space) with ''(empty) ex:"Math
        temp +=j.strip()+" "#" abc ".strip() will return "abc", remove the trailing spaces
        temp = temp.replace('&','_')
    sub_cat_list.append(temp.strip())

project_data['clean_subcategories'] = sub_cat_list
project_data.drop(['project_subject_subcategories'], axis=1, inplace=True)

# count of all the words in corpus python: https://stackoverflow.com/a/22898595/4084039
my_counter = Counter()
for word in project_data['clean_subcategories'].values:
    my_counter.update(word.split())

sub_cat_dict = dict(my_counter)
sorted_sub_cat_dict = dict(sorted(sub_cat_dict.items(), key=lambda kv: kv[1]))
```

1.4 Text preprocessing

1.4.1 Essay Text

In [0]:

```
# merge two column text dataframe:
project_data["essay"] = project_data["project_essay_1"].map(str) +\
                        project_data["project_essay_2"].map(str) + \
                        project_data["project_essay_3"].map(str) + \
                        project_data["project_essay_4"].map(str)
```

In [0]:

```
project_data.head(2)
```

Out[23]:

	Unnamed: 0	id	teacher_id	teacher_prefix	school_stat
0	160221	p253737	c90749f5d961ff158d4b4d1e7dc665fc	Mrs.	IN
1	140945	p258326	897464ce9ddc600bced1151f324dd63a	Mr.	FL

In [0]:

```
type(project_data['project_is_approved'][0])
```

Out[24]:

```
numpy.int64
```

In [0]:

```
# printing some random essays.
print(project_data['essay'].values[0])
print("="*50)
print(project_data['essay'].values[150])
print("="*50)
print(project_data['essay'].values[1000])
print("="*50)
print(project_data['essay'].values[20000])
print("="*50)
print(project_data['essay'].values[99999])
print("="*50)
```

```
My students are English learners that are working on English as their second
or third languages. We are a melting pot of refugees, immigrants, and native
-born Americans bringing the gift of language to our school. \r\n\r\n We hav
e over 24 languages represented in our English Learner program with students
at every level of mastery.  We also have over 40 countries represented with
the families within our school.  Each student brings a wealth of knowledge a
nd experiences to us that open our eyes to new cultures, beliefs, and respec
t.\"The limits of your language are the limits of your world.\"-Ludwig Wittg
enstein  Our English learner's have a strong support system at home that beg
s for more resources.  Many times our parents are learning to read and speak
English along side of their children.  Sometimes this creates barriers for p
arents to be able to help their child learn phonetics, letter recognition, a
nd other reading skills.\r\n\r\nBy providing these dvd's and players, studen
ts are able to continue their mastery of the English language even if no one
at home is able to assist.  All families with students within the Level 1 pr
oficiency status, will be a offered to be a part of this program.  These edu
cational videos will be specially chosen by the English Learner Teacher and
will be sent home regularly to watch.  The videos are to help the child deve
lop early reading skills.\r\n\r\nParents that do not have access to a dvd pl
ayer will have the opportunity to check out a dvd player to use for the yea
r.  The plan is to use these videos and educational dvd's for the years to c
ome for other EL students.\r\nnannan
==================================================
The 51 fifth grade students that will cycle through my classroom this year a
ll love learning, at least most of the time. At our school, 97.3% of the stu
dents receive free or reduced price lunch. Of the 560 students, 97.3% are mi
nority students. \r\nThe school has a vibrant community that loves to get to
gether and celebrate. Around Halloween there is a whole school parade to sho
w off the beautiful costumes that students wear. On Cinco de Mayo we put on
a big festival with crafts made by the students, dances, and games. At the e
nd of the year the school hosts a carnival to celebrate the hard work put in
during the school year, with a dunk tank being the most popular activity.My
students will use these five brightly colored Hokki stools in place of regul
ar, stationary, 4-legged chairs. As I will only have a total of ten in the c
lassroom and not enough for each student to have an individual one, they wil
l be used in a variety of ways. During independent reading time they will be
used as special chairs students will each use on occasion. I will utilize th
em in place of chairs at my small group tables during math and reading time
s. The rest of the day they will be used by the students who need the highes
t amount of movement in their life in order to stay focused on school.\r\n\r
\nWhenever asked what the classroom is missing, my students always say more
Hokki Stools. They can't get their fill of the 5 stools we already have. Whe
n the students are sitting in group with me on the Hokki Stools, they are al
ways moving, but at the same time doing their work. Anytime the students get
to pick where they can sit, the Hokki Stools are the first to be taken. Ther
e are always students who head over to the kidney table to get one of the st
```

```
ools who are disappointed as there are not enough of them. \r\n\r\nWe ask a
lot of students to sit for 7 hours a day. The Hokki stools will be a comprom
ise that allow my students to do desk work and move at the same time. These
stools will help students to meet their 60 minutes a day of movement by allo
wing them to activate their core muscles for balance while they sit. For man
y of my students, these chairs will take away the barrier that exists in sch
ools for a child who can't sit still.nannan
==================================================
How do you remember your days of school? Was it in a sterile environment wit
h plain walls, rows of desks, and a teacher in front of the room? A typical
day in our room is nothing like that. I work hard to create a warm inviting
themed room for my students look forward to coming to each day.\r\n\r\nMy cl
ass is made up of 28 wonderfully unique boys and girls of mixed races in Ark
ansas.\r\nThey attend a Title I school, which means there is a high enough p
ercentage of free and reduced-price lunch to qualify. Our school is an \"ope
n classroom\" concept, which is very unique as there are no walls separating
the classrooms. These 9 and 10 year-old students are very eager learners; th
ey are like sponges, absorbing all the information and experiences and keep
on wanting more.With these resources such as the comfy red throw pillows and
the whimsical nautical hanging decor and the blue fish nets, I will be able
to help create the mood in our classroom setting to be one of a themed nauti
cal environment. Creating a classroom environment is very important in the s
uccess in each and every child's education. The nautical photo props will be
used with each child as they step foot into our classroom for the first time
on Meet the Teacher evening. I'll take pictures of each child with them, hav
e them developed, and then hung in our classroom ready for their first day o
f 4th grade.  This kind gesture will set the tone before even the first day
of school! The nautical thank you cards will be used throughout the year by
the students as they create thank you cards to their team groups.\r\n\r\nYou
r generous donations will help me to help make our classroom a fun, invitin
g, learning environment from day one.\r\n\r\nIt costs lost of money out of m
y own pocket on resources to get our classroom ready. Please consider helpin
g with this project to make our new school year a very successful one. Thank
you!nannan
==================================================
My kindergarten students have varied disabilities ranging from speech and la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. I teach in a Title I school where most of the students receive free o
r reduced price lunch.  Despite their disabilities and limitations, my stude
nts love coming to school and come eager to learn and explore.Have you ever
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble to move as they learn or so they say.Wobble chairs are the answer and I
love then because they develop their core, which enhances gross motor and in
Turn fine motor skills. \r\nThey also want to learn through games, my kids d
on't want to sit and do worksheets. They want to learn to count by jumping a
nd playing. Physical engagement is the key to our success. The number toss a
nd color and shape mats can make that happen. My students will forget they a
re doing work and just have the fun a 6 year old deserves.nannan
==================================================
The mediocre teacher tells. The good teacher explains. The superior teacher
demonstrates. The great teacher inspires. -William A. Ward\r\n\r\nMy school
has 803 students which is makeup is 97.6% African-American, making up the la
rgest segment of the student body. A typical school in Dallas is made up of
23.2% African-American students. Most of the students are on free or reduced
lunch. We aren't receiving doctors, lawyers, or engineers children from rich
backgrounds or neighborhoods. As an educator I am inspiring minds of young c
hildren and we focus not only on academics but one smart, effective, efficie
nt, and disciplined students with good character.In our classroom we can uti
```

```
lize the Bluetooth for swift transitions during class. I use a speaker which
doesn't amplify the sound enough to receive the message. Due to the volume o
f my speaker my students can't hear videos or books clearly and it isn't mak
ing the lessons as meaningful. But with the bluetooth speaker my students wi
ll be able to hear and I can stop, pause and replay it at any time.\r\nThe c
art will allow me to have more room for storage of things that are needed fo
r the day and has an extra part to it I can use.  The table top chart has al
l of the letter, words and pictures for students to learn about different le
tters and it is more accessible.nannan
==================================================
```

In [0]:

```
# https://stackoverflow.com/a/47091490/4084039
import re

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase
```

In [0]:

```
sent = decontracted(project_data['essay'].values[20000])
print(sent)
print("="*50)
```

```
My kindergarten students have varied disabilities ranging from speech and la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations. \r\n\r\nThe materials we have are the ones I seek out for my stu
dents. I teach in a Title I school where most of the students receive free o
r reduced price lunch.  Despite their disabilities and limitations, my stude
nts love coming to school and come eager to learn and explore.Have you ever
felt like you had ants in your pants and you needed to groove and move as yo
u were in a meeting? This is how my kids feel all the time. The want to be a
ble to move as they learn or so they say.Wobble chairs are the answer and I
love then because they develop their core, which enhances gross motor and in
Turn fine motor skills. \r\nThey also want to learn through games, my kids d
o not want to sit and do worksheets. They want to learn to count by jumping
and playing. Physical engagement is the key to our success. The number toss
and color and shape mats can make that happen. My students will forget they
are doing work and just have the fun a 6 year old deserves.nannan
==================================================
```

In [0]:

```
# \r \n \t remove from string python: http://texthandler.com/info/remove-line-breaks-python
sent = sent.replace('\\r', ' ')
sent = sent.replace('\\"', ' ')
sent = sent.replace('\\n', ' ')
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and la
nguage delays, cognitive delays, gross/fine motor delays, to autism. They ar
e eager beavers and always strive to work their hardest working past their l
imitations.     The materials we have are the ones I seek out for my student
s. I teach in a Title I school where most of the students receive free or re
duced price lunch.  Despite their disabilities and limitations, my students 
love coming to school and come eager to learn and explore.Have you ever felt
like you had ants in your pants and you needed to groove and move as you wer
e in a meeting? This is how my kids feel all the time. The want to be able t
o move as they learn or so they say.Wobble chairs are the answer and I love 
then because they develop their core, which enhances gross motor and in Turn
fine motor skills.   They also want to learn through games, my kids do not w
ant to sit and do worksheets. They want to learn to count by jumping and pla
ying. Physical engagement is the key to our success. The number toss and col
or and shape mats can make that happen. My students will forget they are doi
ng work and just have the fun a 6 year old deserves.nannan
```

In [0]:

```
#remove spacial character: https://stackoverflow.com/a/5843547/4084039
sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
print(sent)
```

```
My kindergarten students have varied disabilities ranging from speech and la
nguage delays cognitive delays gross fine motor delays to autism They are ea
ger beavers and always strive to work their hardest working past their limit
ations The materials we have are the ones I seek out for my students I teach
in a Title I school where most of the students receive free or reduced price
lunch Despite their disabilities and limitations my students love coming to 
school and come eager to learn and explore Have you ever felt like you had a
nts in your pants and you needed to groove and move as you were in a meeting
This is how my kids feel all the time The want to be able to move as they le
arn or so they say Wobble chairs are the answer and I love then because they
develop their core which enhances gross motor and in Turn fine motor skills 
They also want to learn through games my kids do not want to sit and do work
sheets They want to learn to count by jumping and playing Physical engagemen
t is the key to our success The number toss and color and shape mats can mak
e that happen My students will forget they are doing work and just have the 
fun a 6 year old deserves nannan
```

In [0]:

```
# https://gist.github.com/sebleier/554280
# we are removing the words from the stop words list: 'no', 'nor', 'not'
stopwords= ['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "
            "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his
            'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they'
            'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'l
            'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had',
            'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'u
            'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'd
            'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over',
            'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', '
            'most', 'other', 'some', 'such', 'only', 'own', 'same', 'so', 'than', 'too', 'v
            's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now',
            've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'do
            "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn',
            "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn'
            'won', "won't", 'wouldn', "wouldn't"]
```

In [0]:

```
# Combining all the above statemennts
from tqdm import tqdm
preprocessed_essays = []
# tqdm is for printing the status bar
for sentance in tqdm(project_data['essay'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_essays.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [01:07<00:00, 1624.89it/s]
```

In [0]:

```
# after preprocesing
preprocessed_essays[20000]
cleaned_project_data = pd.DataFrame()
# project_data['cleaned_essay'] = preprocessed_essays
cleaned_project_data['id'] = project_data['id']
cleaned_project_data['cleaned_essay'] = preprocessed_essays
cleaned_project_data.head(2)
```

Out[32]:

	id	cleaned_essay
0	p253737	my students english learners working english s...
1	p258326	our students arrive school eager learn they po...

1.4.2 Project title Text

In [0]:

```
# printing some random essays.
print(project_data['project_title'].values[0])
print("="*50)
print(project_data['project_title'].values[150])
print("="*50)
print(project_data['project_title'].values[1000])
print("="*50)
print(project_data['project_title'].values[20000])
print("="*50)
print(project_data['project_title'].values[99999])
print("="*50)
```

```
Educational Support for English Learners at Home
==================================================
More Movement with Hokki Stools
==================================================
Sailing Into a Super 4th Grade Year
==================================================
We Need To Move It While We Input It!
==================================================
Inspiring Minds by Enhancing the Educational Experience
==================================================
```

In [0]:

```
# similarly you can preprocess the titles also
preprocessed_titles = []
for sentance in tqdm(project_data['project_title'].values):
    sent = decontracted(sentance)
    sent = sent.replace('\\r', ' ')
    sent = sent.replace('\\"', ' ')
    sent = sent.replace('\\n', ' ')
    sent = re.sub('[^A-Za-z0-9]+', ' ', sent)
    # https://gist.github.com/sebleier/554280
    sent = ' '.join(e for e in sent.split() if e not in stopwords)
    preprocessed_titles.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:03<00:00, 33600.34it/s]
```

In [0]:

```
# similarly you can preprocess the project grade also
preprocessed_grades = []
for sentance in tqdm(project_data['project_grade_category'].values):
#   sent = decontracted(sentance)
#   sent = sent.replace('\\r', ' ')
#   sent = sent.replace('\\"', ' ')
#   sent = sent.replace('\\n', ' ')
  sent = re.sub('[^A-Za-z0-9]+', '_', sentance)
  # https://gist.github.com/sebleier/554280
  sent = ' '.join(e for e in sent.split() if e not in stopwords)
  preprocessed_grades.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:00<00:00, 164717.36it/s]
```

In [0]:

```
print(set(preprocessed_grades))
```

```
{'grades_prek_2', 'grades_6_8', 'grades_9_12', 'grades_3_5'}
```

In [0]:

```
# similarly you can preprocess the teacher_prefix also
preprocessed_teacher_prefix = []
for sentance in tqdm(project_data['teacher_prefix'].values):
  sent = str(sentance).replace('.', '')
#   sent = sent.replace('\\"', ' ')
#   sent = sent.replace('\\n', ' ')
#   sent = re.sub('[^A-Za-z0-9]+', '', sentance)
  # https://gist.github.com/sebleier/554280
  sent = ' '.join(e for e in sent.split() if e not in stopwords)
  preprocessed_teacher_prefix.append(sent.lower().strip())
```

```
100%|██████████| 109248/109248 [00:00<00:00, 198188.06it/s]
```

In [0]:

```
print(preprocessed_teacher_prefix)
```

```
['mrs', 'mr', 'ms', 'mrs', 'mrs', 'mrs', 'mrs', 'ms', 'mrs', 'ms', 'mrs',
'ms', 'mrs', 'mrs', 'ms', 'ms', 'mrs', 'ms', 'mrs', 'ms', 'mrs', 'mrs', 'm
s', 'mr', 'mrs', 'mrs', 'ms', 'teacher', 'mrs', 'mrs', 'mrs', 'mrs', 'mr
s', 'mrs', 'ms', 'mr', 'mrs', 'ms', 'mrs', 'ms', 'ms', 'mrs', 'ms', 'mrs',
'mrs', 'mrs', 'ms', 'mrs', 'ms', 'mrs', 'mrs', 'ms', 'mr', 'ms', 'ms', 'mr
s', 'ms', 'mrs', 'mrs', 'mr', 'ms', 'mrs', 'mr', 'mr', 'ms', 'ms', 'mrs',
'ms', 'ms', 'ms', 'mrs', 'mrs', 'ms', 'mrs', 'mrs', 'teacher', 'ms', 'mr
s', 'mrs', 'ms', 'ms', 'mrs', 'teacher', 'ms', 'mrs', 'ms', 'ms', 'ms', 't
eacher', 'ms', 'mrs', 'mrs', 'ms', 'ms', 'mrs', 'ms', 'ms', 'mrs', 'mrs',
'mrs', 'ms', 'mrs', 'ms', 'mr', 'ms', 'mr', 'mrs', 'mrs', 'mrs', 'ms', 'm
s', 'mrs', 'mrs', 'mrs', 'mrs', 'mrs', 'ms', 'mrs', 'mrs', 'mrs', 'mrs',
'ms', 'ms', 'mrs', 'mrs', 'mrs', 'mrs', 'mrs', 'mr', 'mrs', 'mrs', 'mrs',
'ms', 'mrs', 'mrs', 'mr', 'ms', 'mrs', 'mrs', 'mrs', 'ms', 'ms', 'ms', 'mr
s', 'mrs', 'mrs', 'mrs', 'mrs', 'ms', 'mrs', 'ms', 'ms', 'mrs', 'mrs', 'm
s', 'mrs', 'ms', 'mrs', 'mrs', 'mr', 'mrs', 'mrs', 'ms', 'ms', 'mr', 'ms',
'mr', 'mr', 'mr', 'mrs', 'mrs', 'mrs', 'ms', 'mrs', 'mrs', 'mrs', 'mrs',
'mrs', 'ms', 'ms', 'mrs', 'ms', 'mrs', 'mr', 'ms', 'mrs', 'mrs', 'mrs', 'm
s', 'mrs', 'mrs', 'mrs', 'mr', 'mrs', 'ms', 'mrs', 'mr', 'mrs', 'mrs', 'mr
s', 'ms', 'teacher', 'ms', 'mrs', 'mrs', 'ms', 'mrs', 'mrs', 'mrs', 'mr',
```

In [0]:

```
# project_data['cleaned_project_title'] = preprocessed_titles
cleaned_project_data['cleaned_project_title'] = preprocessed_titles
cleaned_project_data['clean_categories'] = project_data['clean_categories']
cleaned_project_data['clean_subcategories'] = project_data['clean_subcategories']
cleaned_project_data['project_is_approved'] = project_data['project_is_approved']
cleaned_project_data['teacher_prefix'] = preprocessed_teacher_prefix
cleaned_project_data['school_state'] = project_data['school_state']
cleaned_project_data['project_grade_category'] = preprocessed_grades
cleaned_project_data['teacher_number_of_previously_posted_projects'] = project_data['teache
# cleaned_project_data.head(5)
cleaned_project_data.to_csv(os.path.join(dir_path,'cleaned_project_data.csv'))
```

In [0]:

```
cleaned_project_data.head(2)
```

Out[40]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategori
0	p253737	my students english learners working english s...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

In [0]:

```
cleaned_project_data['cleaned_text'] = cleaned_project_data['cleaned_essay'].map(str) + \
                                        cleaned_project_data['cleaned_project_title'].map(s
cleaned_project_data.head(2)
```

Out[41]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategori
0	p253737	my students english learners working english s...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

In [0]:

```
set(cleaned_project_data['teacher_prefix'])
```

Out[42]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

Computing Sentiment Scores

In [0]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

# import nltk
# nltk.download('vader_lexicon')

sid = SentimentIntensityAnalyzer()

for_sentiment = 'a person is a person no matter how small dr seuss i teach the smallest stu
for learning my students learn in many different ways using all of our senses and multiple 
of techniques to help all my students succeed students in my class come from a variety of d
for wonderful sharing of experiences and cultures including native americans our school is 
learners which can be seen through collaborative student project based learning in and out 
in my class love to work with hands on materials and have many different opportunities to p
mastered having the social skills to work cooperatively with friends is a crucial aspect of
montana is the perfect place to learn about agriculture and nutrition my students love to r
in the early childhood classroom i have had several kids ask me can we try cooking with rea
and create common core cooking lessons where we learn important math and writing concepts w
food for snack time my students will have a grounded appreciation for the work that went in
of where the ingredients came from as well as how it is healthy for their bodies this proje
nutrition and agricultural cooking recipes by having us peel our own apples to make homemad
and mix up healthy plants from our classroom garden in the spring we will also create our c
shared with families students will gain math and literature skills as well as a life long e
nannan'
ss = sid.polarity_scores(for_sentiment)

for k in ss:
    print('{0}: {1}, '.format(k, ss[k]), end='')

# we can use these 4 things as features/attributes (neg, neu, pos, compound)
# neg: 0.0, neu: 0.753, pos: 0.247, compound: 0.93
```

```
neg: 0.01, neu: 0.745, pos: 0.245, compound: 0.9975, 
```

Tasks: Logistic Regression

1. **[Task-1] Logistic Regression(either SGDClassifier with log loss, or LogisticRegression) on these feature sets**

    - Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (`BOW with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 2: categorical, numerical features + project_title(TFIDF)+ preprocessed_eassay (`TFIDF with bi-grams` with `min_df=10` and `max_features=5000`)
    - Set 3: categorical, numerical features + project_title(AVG W2V)+ preprocessed_eassay (AVG W2V)
    - Set 4: categorical, numerical features + project_title(TFIDF W2V)+ preprocessed_essay (TFIDF W2V)

2. **Hyper paramter tuning (find best hyper parameters corresponding the algorithm that you choose)**

    - Find the best hyper parameter which will give the maximum AUC (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/receiver-operating-characteristic-curve-roc-curve-and-auc-1/) value

- Find the best hyper paramter using k-fold cross validation or simple cross validation data
- Use gridsearch cv or randomsearch cv or you can also write your own for loops to do this task of hyperparameter tuning

3. **Representation of results**

    - You need to plot the performance of model both on train data and cross validation data for each hyper parameter, like shown in the figure.

    - Once after you found the best hyper parameter, you need to train your model with it, and find the AUC on test data and plot the ROC curve on both train and test.

    - Along with plotting ROC curve, you need to print the confusion matrix (https://www.appliedaicourse.com/course/applied-ai-course-online/lessons/confusion-matrix-tpr-fpr-fnr-tnr-1/) with predicted and original labels of test data points. Please visualize your confusion matrices using seaborn heatmaps.

      (https://seaborn.pydata.org/generated/seaborn.heatmap.html)

4. **[Task-2] Apply Logistic Regression on the below feature set Set 5 by finding the best hyper parameter as suggested in step 2 and step 3.**
5. Consider these set of features Set 5 :

    - **school_state** : categorical data
    - **clean_categories** : categorical data
    - **clean_subcategories** : categorical data
    - **project_grade_category** :categorical data
    - **teacher_prefix** : categorical data
    - **quantity** : numerical data
    - **teacher_number_of_previously_posted_projects** : numerical data
    - **price** : numerical data
    - **sentiment score's of each of the essay** : numerical data
    - **number of words in the title** : numerical data
    - **number of words in the combine essays** : numerical data

    And apply the Logistic regression on these features by finding the best hyper paramter as suggested in step 2 and step 3

6. **Conclusion**

    - You need to summarize the results at the end of the notebook, summarize it in the table format. To print out a table please refer to this prettytable library link (http://zetcode.com/python/prettytable/)

Note: Data Leakage

1. There will be an issue of data-leakage if you vectorize the entire data and then split it into train/cv/test.
2. To avoid the issue of data-leakage, make sure to split your data first and then vectorize it.
3. While vectorizing your data, apply the method fit_transform() on you train data, and apply the method transform() on cv/test data.

4. For more details please go through this link. (https://soundcloud.com/applied-ai-course/leakage-bow-and-tfidf)

2. Logistic Regression

2.1 Splitting data into Train and cross validation(or test): Stratified Sampling

In [0]:

```
# Taking complete data
project_65k = cleaned_project_data[:65000].copy()
project_65k.shape
```

Out[36]:

```
(65000, 11)
```

In [0]:

```
set(project_65k['teacher_prefix'])
```

Out[37]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

In [0]:

```
print(len(project_65k[project_65k['project_is_approved'] == 0]))
print(len(project_65k[project_65k['project_is_approved'] == 1]))
```

```
9895
55105
```

In [0]:

```
X = project_65k.drop(['project_is_approved'], axis=1)
y = project_65k['project_is_approved']
print(type(y))
```

```
<class 'pandas.core.series.Series'>
```

In [0]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [0]:

```
print(y_train.value_counts())
print(len(y_train == 0))
print(len(y_train == 1))
```

```
1    24736
0     4442
Name: project_is_approved, dtype: int64
29178
29178
```

In [0]:

```
test = pd.concat([X_train, y_train], axis=1)
test.head(2)
print(len(test[test['project_is_approved'] == 0]))
print(len(test[test['project_is_approved'] == 1]))
```

```
4442
24736
```

In [0]:

```
from sklearn.utils import resample
```

In [0]:

```
#https://elitedatascience.com/imbalanced-classes
# Separate majority and minority classes
project_majority = test[test.project_is_approved==1]
project_minority = test[test.project_is_approved==0]

# Upsample minority class
project_minority_upsampled = resample(project_minority,
                                 replace=True,     # sample with replacement
                                 n_samples=24736,    # to match majority class
                                 random_state=123) # reproducible results

# Combine majority class with upsampled minority class
project_upsampled = pd.concat([project_majority, project_minority_upsampled])

# Display new class counts
project_upsampled.project_is_approved.value_counts()
```

Out[48]:

```
1    24736
0    24736
Name: project_is_approved, dtype: int64
```

In [0]:

```
X_train = project_upsampled.drop(['project_is_approved'], axis=1)
y_train = project_upsampled['project_is_approved']
```

In [0]:

```
set(X_train['teacher_prefix'])
```

Out[50]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

In [0]:

```
print(y_train.value_counts())
```

```
1    24736
0    24736
Name: project_is_approved, dtype: int64
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/y_train.pkl","wb") as file
  pickle.dump(y_train, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/y_cv.pkl","wb") as file:
  pickle.dump(y_cv, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/y_test.pkl","wb") as file:
  pickle.dump(y_test, file)
```

In [0]:

```
print(len(X_train), len(y_train))
print(len(X_cv), len(y_cv))
print(len(X_test), len(y_test))
```

```
49472 49472
14372 14372
21450 21450
```

In [0]:

```
print(set(X_train['teacher_prefix'].values))
print(set(X_cv['teacher_prefix'].values))
print(set(X_test['teacher_prefix'].values))
```

```
{'nan', 'mrs', 'ms', 'mr', 'dr', 'teacher'}
{'mrs', 'ms', 'mr', 'dr', 'teacher'}
{'nan', 'mrs', 'ms', 'mr', 'dr', 'teacher'}
```

2.2 Make Data Model Ready: encoding numerical, categorical features

Project Category

In [0]:

```
# we use count vectorizer to convert the values into one hot encoded features
# Project Category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bina
X_train_one_hot_clean_cat = vectorizer.fit_transform(X_train['clean_categories'].values)
X_cv_one_hot_clean_cat = vectorizer.transform(X_cv['clean_categories'].values)
X_test_one_hot_clean_cat = vectorizer.transform(X_test['clean_categories'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_cat.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_cat.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_cat.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (49472, 9)
Shape of matrix after one hot encodig  (14372, 9)
Shape of matrix after one hot encodig  (21450, 9)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_one_hot_clean_cat.
  pickle.dump(X_train_one_hot_clean_cat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_one_hot_clean_cat.pkl
  pickle.dump(X_cv_one_hot_clean_cat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_one_hot_clean_cat.p
  pickle.dump(X_test_one_hot_clean_cat, file)
```

Project Sub-category

In [0]:

```
# Project Sub-category
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
X_train_one_hot_clean_sub_cat = vectorizer.fit_transform(X_train['clean_subcategories'].val
X_cv_one_hot_clean_sub_cat = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_one_hot_clean_sub_cat = vectorizer.transform(X_test['clean_subcategories'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_sub_cat.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_sub_cat.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_sub_cat.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_
Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness',
'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (49472, 30)
Shape of matrix after one hot encodig  (14372, 30)
Shape of matrix after one hot encodig  (21450, 30)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_one_hot_clean_sub_
  pickle.dump(X_train_one_hot_clean_sub_cat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_one_hot_clean_sub_cat
  pickle.dump(X_cv_one_hot_clean_sub_cat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_one_hot_clean_sub_c
  pickle.dump(X_test_one_hot_clean_sub_cat, file)
```

School State

In [0]:

```
# School State
vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_school_state = vectorizer.fit_transform(X_train['school_state'].value
X_cv_one_hot_clean_school_state = vectorizer.transform(X_cv['school_state'].values)
X_test_one_hot_clean_school_state = vectorizer.transform(X_test['school_state'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_school_state.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_school_state.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_school_state.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'I
A', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'O
R', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV',
'WY']
Shape of matrix after one hot encodig  (49472, 51)
Shape of matrix after one hot encodig  (14372, 51)
Shape of matrix after one hot encodig  (21450, 51)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_one_hot_clean_scho
  pickle.dump(X_train_one_hot_clean_school_state, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_one_hot_clean_school_
  pickle.dump(X_cv_one_hot_clean_school_state, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_one_hot_clean_scho
  pickle.dump(X_test_one_hot_clean_school_state, file)
```

Teacher Prefix

In [0]:

```
# Teacher Prefix
import re
X_train_prefix_list = []
X_cv_prefix_list = []
X_test_prefix_list = []
for s in tqdm(X_train['teacher_prefix'].values):
    train_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    train_prefix = re.sub('nan', '', str(train_prefix))
    X_train_prefix_list.append(train_prefix)
for s in tqdm(X_cv['teacher_prefix'].values):
    cv_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    X_cv_prefix_list.append(cv_prefix)
for s in tqdm(X_test['teacher_prefix'].values):
    test_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    test_prefix = re.sub('nan', '', str(test_prefix))
    X_test_prefix_list.append(test_prefix)

vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_teacher_prefix = vectorizer.fit_transform(X_train_prefix_list)
X_cv_one_hot_clean_teacher_prefix = vectorizer.transform(X_cv_prefix_list)
X_test_one_hot_clean_teacher_prefix = vectorizer.fit_transform(X_test_prefix_list)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_teacher_prefix.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_teacher_prefix.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_teacher_prefix.shape)
```

```
100%|██████████| 49472/49472 [00:00<00:00, 293375.40it/s]
100%|██████████| 14372/14372 [00:00<00:00, 465418.49it/s]
100%|██████████| 21450/21450 [00:00<00:00, 293472.54it/s]

['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig  (49472, 5)
Shape of matrix after one hot encodig  (14372, 5)
Shape of matrix after one hot encodig  (21450, 5)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_one_hot_clean_teac
  pickle.dump(X_train_one_hot_clean_teacher_prefix, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_one_hot_clean_teacher
  pickle.dump(X_cv_one_hot_clean_teacher_prefix, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_one_hot_clean_teach
  pickle.dump(X_test_one_hot_clean_teacher_prefix, file)
```

Project Grade Category

In [0]:

```
# Project Grade Category

def grade_cat_cleaning(data):
  proj_grade_cat_list = []
  for grade in tqdm(data):
#       grade_cat = re.sub('-',' to ', grade)
#       grade_cat = re.sub('2',' two ', grade_cat)
#       grade_cat = re.sub('3',' three ', grade_cat)
#       grade_cat = re.sub('5',' five ', grade_cat)
#       grade_cat = re.sub('6',' six ', grade_cat)
#       grade_cat = re.sub('8',' eight ', grade_cat)
#       grade_cat = re.sub('9',' nine ', grade_cat)
#       grade_cat = re.sub('12',' twelve ', grade_cat)
      proj_grade_cat_list.append(grade.lower().strip())
  return proj_grade_cat_list
X_train_proj_grade_cat = grade_cat_cleaning([sent for sent in X_train['project_grade_catego
X_cv_proj_grade_cat = grade_cat_cleaning([sent for sent in X_cv['project_grade_category'].v
X_test_proj_grade_cat = grade_cat_cleaning([sent for sent in X_test['project_grade_category
vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_project_grade = vectorizer.fit_transform(X_train_proj_grade_cat)
X_cv_one_hot_clean_project_grade = vectorizer.transform(X_cv_proj_grade_cat)
X_test_one_hot_clean_project_grade = vectorizer.transform(X_test_proj_grade_cat)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_project_grade.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_project_grade.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_project_grade.shape)
```

```
100%|██████████| 49472/49472 [00:00<00:00, 1015850.19it/s]
100%|██████████| 14372/14372 [00:00<00:00, 1199779.81it/s]
100%|██████████| 21450/21450 [00:00<00:00, 1139915.37it/s]

['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encodig  (49472, 4)
Shape of matrix after one hot encodig  (14372, 4)
Shape of matrix after one hot encodig  (21450, 4)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_one_hot_clean_proj
  pickle.dump(X_train_one_hot_clean_project_grade, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_one_hot_clean_project
  pickle.dump(X_cv_one_hot_clean_project_grade, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_one_hot_clean_proje
  pickle.dump(X_test_one_hot_clean_project_grade, file)
```

2.3 Make Data Model Ready: encoding eassay, and project_title

In [0]:

```
X_train.head(2)
```

Out[65]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcate
25457	p050945	our rural school high poverty area many studen...	wiggle while you work	SpecialNeeds	SpecialNeeds
22688	p251489	my students terrific personalities try hard lo...	books books books	Literacy_Language	Literacy

2.3.1: BOW

Project Title

In [0]:

```
count_vectorizer = CountVectorizer(min_df=10)
X_train_bow_project_title = count_vectorizer.fit_transform(X_train['cleaned_project_title']
X_cv_bow_project_title = count_vectorizer.transform(X_cv['cleaned_project_title'])
X_test_bow_project_title = count_vectorizer.transform(X_test['cleaned_project_title'])
print("Shape of matrix after BOW encodig ",X_train_bow_project_title.shape)
print("Shape of matrix after BOW encodig ",X_cv_bow_project_title.shape)
print("Shape of matrix after BOW encodig ",X_test_bow_project_title.shape)
```

```
Shape of matrix after BOW encodig  (49472, 2210)
Shape of matrix after BOW encodig  (14372, 2210)
Shape of matrix after BOW encodig  (21450, 2210)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_bow_project_title.
  pickle.dump(X_train_bow_project_title, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_bow_project_title.pkl
  pickle.dump(X_cv_bow_project_title, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_bow_project_title.p
  pickle.dump(X_test_bow_project_title, file)
```

Essays

In [0]:

```
count_vectorizer = CountVectorizer(min_df=10, ngram_range = (1,2), max_features = 5000)
X_train_bow_essays = count_vectorizer.fit_transform(X_train['cleaned_essay'])
X_cv_bow_essays = count_vectorizer.transform(X_cv['cleaned_essay'])
X_test_bow_essays = count_vectorizer.transform(X_test['cleaned_essay'])
print("Shape of matrix after BOW encodig ",X_train_bow_essays.shape)
print("Shape of matrix after BOW encodig ",X_cv_bow_essays.shape)
print("Shape of matrix after BOW encodig ",X_test_bow_essays.shape)
```

```
Shape of matrix after BOW encodig  (49472, 5000)
Shape of matrix after BOW encodig  (14372, 5000)
Shape of matrix after BOW encodig  (21450, 5000)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_bow_essays.pkl","w
  pickle.dump(X_train_bow_essays, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_bow_essays.pkl","wb")
  pickle.dump(X_cv_bow_essays, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_bow_essays.pkl","wb
  pickle.dump(X_test_bow_essays, file)
```

2.3.2: TFIDF Vectorizer

Project Title

In [0]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vectorizer = TfidfVectorizer(min_df=10)
X_train_tfidf_project_title = tfidf_vectorizer.fit_transform(X_train['cleaned_project_title
X_cv_tfidf_project_title = tfidf_vectorizer.transform(X_cv['cleaned_project_title'])
X_test_tfidf_project_title = tfidf_vectorizer.transform(X_test['cleaned_project_title'])
print("Shape of matrix after TFIDF encodig ",X_train_tfidf_project_title.shape)
print("Shape of matrix after TFIDF encodig ",X_cv_tfidf_project_title.shape)
print("Shape of matrix after TFIDF encodig ",X_test_tfidf_project_title.shape)
```

```
Shape of matrix after TFIDF encodig  (49472, 2210)
Shape of matrix after TFIDF encodig  (14372, 2210)
Shape of matrix after TFIDF encodig  (21450, 2210)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_tfidf_project_titl
  pickle.dump(X_train_tfidf_project_title, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_tfidf_project_title.p
  pickle.dump(X_cv_tfidf_project_title, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_tfidf_project_title
  pickle.dump(X_test_tfidf_project_title, file)
```

Essays

In [0]:

```
tfidf_vectorizer = TfidfVectorizer(min_df=10, ngram_range = (1,2), max_features = 5000)
X_train_tfidf_essays = tfidf_vectorizer.fit_transform(X_train['cleaned_essay'])
X_cv_tfidf_essays = tfidf_vectorizer.transform(X_cv['cleaned_essay'])
X_test_tfidf_essays = tfidf_vectorizer.transform(X_test['cleaned_essay'])
print("Shape of matrix after TFIDF encodig ",X_train_tfidf_essays.shape)
print("Shape of matrix after TFIDF encodig ",X_cv_tfidf_essays.shape)
print("Shape of matrix after TFIDF encodig ",X_test_tfidf_essays.shape)
```

```
Shape of matrix after TFIDF encodig  (49472, 5000)
Shape of matrix after TFIDF encodig  (14372, 5000)
Shape of matrix after TFIDF encodig  (21450, 5000)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_tfidf_essays.pkl",
  pickle.dump(X_train_tfidf_essays, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_tfidf_essays.pkl","wb
  pickle.dump(X_cv_tfidf_essays, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_tfidf_essays.pkl","
  pickle.dump(X_test_tfidf_essays, file)
```

2.3.3: Avg W2V

Cleaned Text

In [0]:

```
from gensim.models import Word2Vec
from gensim.models import KeyedVectors
```

In [0]:

```
X_train_list_of_sent=[]
X_cv_list_of_sent=[]
X_test_list_of_sent=[]
for sent in X_train['cleaned_text'].values:
    X_train_list_of_sent.append(sent.split())
for sent in X_cv['cleaned_text'].values:
    X_cv_list_of_sent.append(sent.split())
for sent in X_test['cleaned_text'].values:
    X_test_list_of_sent.append(sent.split())
```

In [0]:

```
#train data avg w2v
w2v_model=Word2Vec(X_train_list_of_sent,min_count=5,size=50, workers=4)
w2v_words = list(w2v_model.wv.vocab)
```

In [0]:

```
# average Word2Vec
# compute average word2vec for each text.
def avg_w2v(sent_list):
  sent_vectors = []; # the avg-w2v for each sentence is stored in this list
  for sent in tqdm(sent_list): # for each tain sentence
      sent_vec = np.zeros(50) # as word vectors are of zero length
      cnt_words =0; # num of words with a valid vector in the sentence
      for word in sent: # for each word in a sentence
          if word in w2v_words:
              vec = w2v_model.wv[word]
              sent_vec += vec
              cnt_words += 1
      if cnt_words != 0:
          sent_vec /= cnt_words
      sent_vectors.append(sent_vec)
  print("\n",len(sent_vectors))
  print(len(sent_vectors[0]))
  return sent_vectors
```

In [0]:

```
X_train_avg_w2v_text = avg_w2v([sent.split() for sent in X_train['cleaned_text']])
X_cv_avg_w2v_text = avg_w2v([sent.split() for sent in X_cv['cleaned_text']])
X_test_avg_w2v_text = avg_w2v([sent.split() for sent in X_test['cleaned_text']])
print("Shape of matrix after Avg W2V encodig ",len(X_train_avg_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_cv_avg_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_test_avg_w2v_text))
```

```
100%|██████████| 49472/49472 [04:28<00:00, 183.95it/s]
  0%|          | 0/14372 [00:00<?, ?it/s]


 49472
50

100%|██████████| 14372/14372 [01:22<00:00, 175.08it/s]


 14372
50

100%|██████████| 21450/21450 [02:05<00:00, 170.90it/s]


 21450
50
Shape of matrix after Avg W2V encodig  49472
Shape of matrix after Avg W2V encodig  14372
Shape of matrix after Avg W2V encodig  21450
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_avg_w2v_text.pkl",
  pickle.dump(X_train_avg_w2v_text, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_avg_w2v_text.pkl","wb
  pickle.dump(X_cv_avg_w2v_text, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_avg_w2v_text.pkl","
  pickle.dump(X_test_avg_w2v_text, file)
```

2.3.4: TFIDF Weighted W2V

Cleaned Text

In [0]:

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_model = TfidfVectorizer()
X_train_tfidf_w2v_model_text = tfidf_model.fit_transform(X_train['cleaned_text'])
# we are converting a dictionary with word as a key, and the idf as a value
dictionary = dict(zip(tfidf_model.get_feature_names(), list(tfidf_model.idf_)))
tfidf_words = set(tfidf_model.get_feature_names())
```

In [0]:

```
# TF-IDF weighted Word2Vec
tfidf_features = tfidf_model.get_feature_names() # tfidf words/col-names
# final_tf_idf is the sparse matrix with row= sentence, col=word and cell_val = tfidf

def tfidf_w2v(sen_list):
  tfidf_w2v_vectors = []; # the tfidf-w2v for each sentence/review is stored in this list
  row=0;

  for sent in tqdm(sen_list): # for each review/sentence
      vector = np.zeros(50) # as word vectors are of zero length
      tf_idf_weight =0; # num of words with a valid vector in the sentence/review
      for word in sent: # for each word in a review/sentence
          if (word in w2v_words) and (word in tfidf_features):
              vec = w2v_model.wv[word]
              tf_idf = dictionary[word]*(sent.count(word)/len(sent))
              vector += (vec * tf_idf)
              tf_idf_weight += tf_idf
      if tf_idf_weight != 0:
          vector /= tf_idf_weight
      tfidf_w2v_vectors.append(vector)
      row += 1
  return tfidf_w2v_vectors
```

In [0]:

```
X_train_tfidf_w2v_text = tfidf_w2v([sent.split() for sent in X_train['cleaned_text']])
X_cv_tfidf_w2v_text = tfidf_w2v([sent.split() for sent in X_cv['cleaned_text']])
X_test_tfidf_w2v_text = tfidf_w2v([sent.split() for sent in X_test['cleaned_text']])
print("Shape of matrix after Avg W2V encodig ",len(X_train_tfidf_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_cv_tfidf_w2v_text))
print("Shape of matrix after Avg W2V encodig ",len(X_test_tfidf_w2v_text))
```

```
100%|██████████| 49472/49472 [1:08:58<00:00, 12.83it/s]
100%|██████████| 14372/14372 [20:24<00:00, 13.20it/s]
100%|██████████| 21450/21450 [30:24<00:00, 11.76it/s]

Shape of matrix after Avg W2V encodig  49472
Shape of matrix after Avg W2V encodig  14372
Shape of matrix after Avg W2V encodig  21450
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_tfidf_w2v_text.pkl
  pickle.dump(X_train_tfidf_w2v_text, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_tfidf_w2v_text.pkl","
  pickle.dump(X_cv_tfidf_w2v_text, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_tfidf_w2v_text.pkl"
  pickle.dump(X_test_tfidf_w2v_text, file)
```

Merging Categorical and Numerical Features

In [0]:

```
from scipy.sparse import csr_matrix
X_train_no_of_projects = np.array(X_train['teacher_number_of_previously_posted_projects'])
X_train_no_of_projects = csr_matrix(X_train_no_of_projects).T
X_cv_no_of_projects = np.array(X_cv['teacher_number_of_previously_posted_projects'])
X_cv_no_of_projects = csr_matrix(X_cv_no_of_projects).T
X_test_no_of_projects = np.array(X_test['teacher_number_of_previously_posted_projects'])
X_test_no_of_projects = csr_matrix(X_test_no_of_projects).T
print(X_train_no_of_projects.shape)
print(X_cv_no_of_projects.shape)
print(X_test_no_of_projects.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

In [0]:

```
from scipy.sparse import coo_matrix, hstack
print(X_train_one_hot_clean_cat.shape, X_train_one_hot_clean_sub_cat.shape, X_train_one_hot
                                 X_train_one_hot_clean_teacher_prefix.shape, X_train_one_ho
X_train_categorical_numerical = hstack([X_train_one_hot_clean_cat, X_train_one_hot_clean_su
                              X_train_one_hot_clean_teacher_prefix, X_train_one_hot_clean_p
X_cv_categorical_numerical = hstack([X_cv_one_hot_clean_cat, X_cv_one_hot_clean_sub_cat, X_
                                X_cv_one_hot_clean_teacher_prefix, X_cv_one_hot_clean_proje
X_test_categorical_numerical = hstack([X_test_one_hot_clean_cat, X_test_one_hot_clean_sub_c
                                X_test_one_hot_clean_teacher_prefix, X_test_one_hot_clean_p
print(X_train_categorical_numerical.shape, X_cv_categorical_numerical.shape, X_test_categor
```

```
(49472, 9) (49472, 30) (49472, 51) (49472, 5) (49472, 4)
(49472, 100) (14372, 100) (21450, 100)
```

SET 1: Merging: categorical, numerical features + project_title(BOW) + preprocessed_essay (BOW)

In [0]:

```
X_train_bow_feat = hstack([X_train_categorical_numerical, X_train_bow_project_title, X_trai
X_cv_bow_feat = hstack([X_cv_categorical_numerical, X_cv_bow_project_title, X_cv_bow_essays
X_test_bow_feat = hstack([X_test_categorical_numerical, X_test_bow_project_title, X_test_bc
print(X_train_bow_feat.shape)
print(X_cv_bow_feat.shape)
print(X_test_bow_feat.shape)
```

```
(49472, 7310)
(14372, 7310)
(21450, 7310)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_bow_feat.pkl","wb"
  pickle.dump(X_train_bow_feat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_bow_feat.pkl","wb") a
  pickle.dump(X_cv_bow_feat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_bow_feat.pkl","wb")
  pickle.dump(X_test_bow_feat, file)
```

SET 2: Merging: categorical, numerical features + project_title(TFIDF) + preprocessed_essay (TFIDF)

In [0]:

```
X_train_tfidf_feat = hstack([X_train_categorical_numerical, X_train_tfidf_project_title, X_
X_cv_tfidf_feat = hstack([X_cv_categorical_numerical, X_cv_tfidf_project_title, X_cv_tfidf_
X_test_tfidf_feat = hstack([X_test_categorical_numerical, X_test_tfidf_project_title, X_tes
print(X_train_tfidf_feat.shape)
print(X_cv_tfidf_feat.shape)
print(X_test_tfidf_feat.shape)
```

```
(49472, 7310)
(14372, 7310)
(21450, 7310)
```

In [0]:

```
with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_train_tfidf_feat.pkl","w
  pickle.dump(X_train_tfidf_feat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_cv_tfidf_feat.pkl","wb")
  pickle.dump(X_cv_tfidf_feat, file)

with open("/content/drive/My Drive/appliedai/Logistic_Regression/X_test_tfidf_feat.pkl","wb
  pickle.dump(X_test_tfidf_feat, file)
```

2.4 Appling Logistic Regression on different kind of featurization as mentioned in the instructions

Apply Logistic Regression on different kind of featurization as mentioned in the instructions
For Every model that you work on make sure you do the step 2 and step 3 of instrucations

Set 1: categorical, numerical features + project_title(BOW) + preprocessed_eassay (BOW with bi-grams with min_df=10 and max_features=5000)

In [0]:

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
```

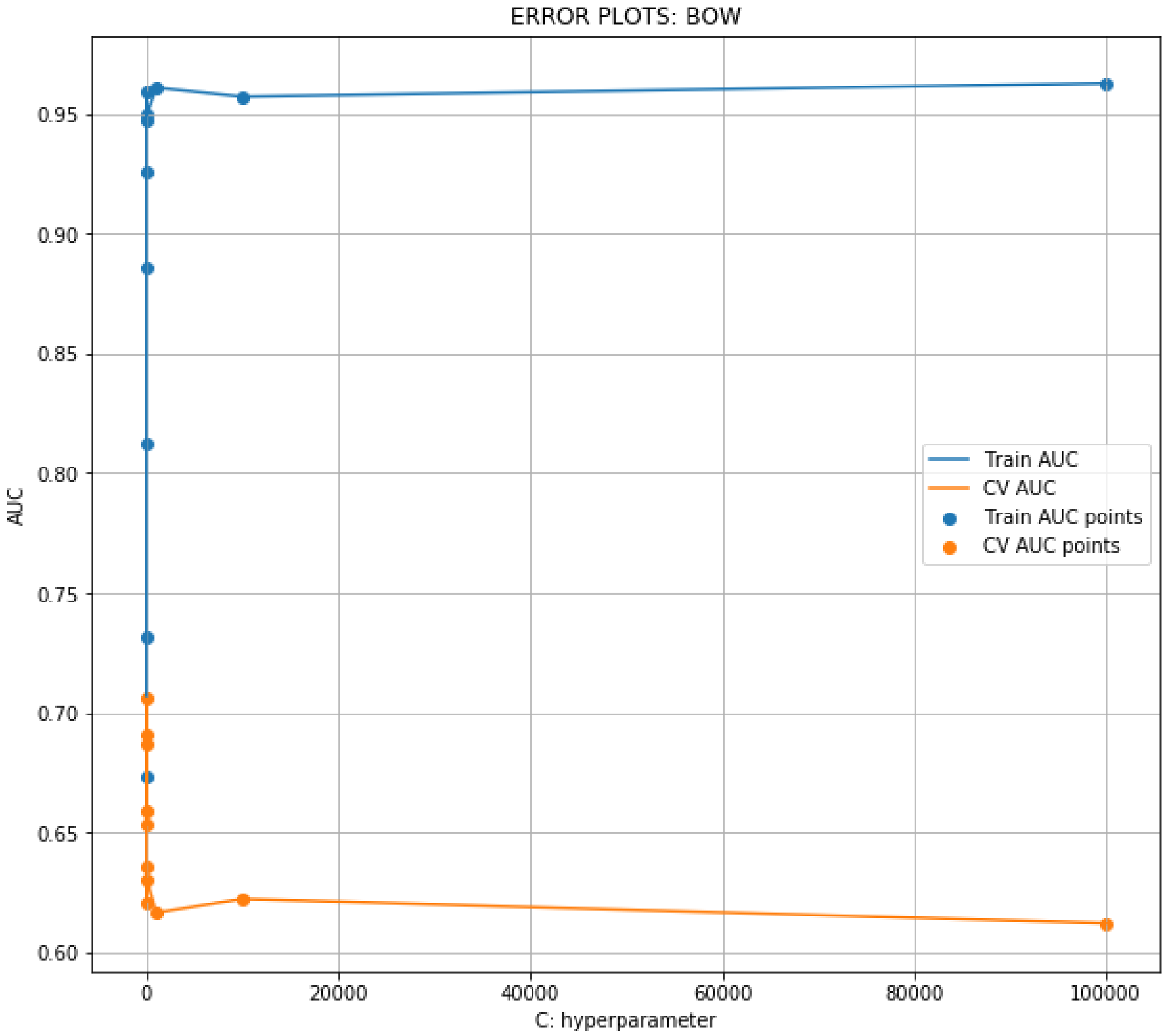
In [0]:

```
# Creating list of C for LR
c_bow = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
print(c_bow)
train_auc_bow = []
cv_auc_bow = []
for c in tqdm(c_bow):
    lr_bow = LogisticRegression(C=c)
    lr_bow.fit(X_train_bow_feat, y_train)

    #predict probabilities for train and validation
    y_train_pred_bow = lr_bow.predict_proba(X_train_bow_feat)[:,1]
    y_cv_pred_bow = lr_bow.predict_proba(X_cv_bow_feat)[:,1]

#     y_train_pred = batch_predict(neigh, X_tr)
#     y_cv_pred = batch_predict(neigh, X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    train_auc_bow.append(roc_auc_score(y_train,y_train_pred_bow))
    cv_auc_bow.append(roc_auc_score(y_cv, y_cv_pred_bow))
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

[1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]

100%|██████████| 11/11 [11:29<00:00, 120.58s/it]
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(c_bow, train_auc_bow, label='Train AUC')
plt.plot(c_bow, cv_auc_bow, label='CV AUC')

plt.scatter(c_bow, train_auc_bow, label='Train AUC points')
plt.scatter(c_bow, cv_auc_bow, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
best_c = c_bow[cv_auc_bow.index(max(cv_auc_bow))]
print(best_c)
print(max(cv_auc_bow))
```

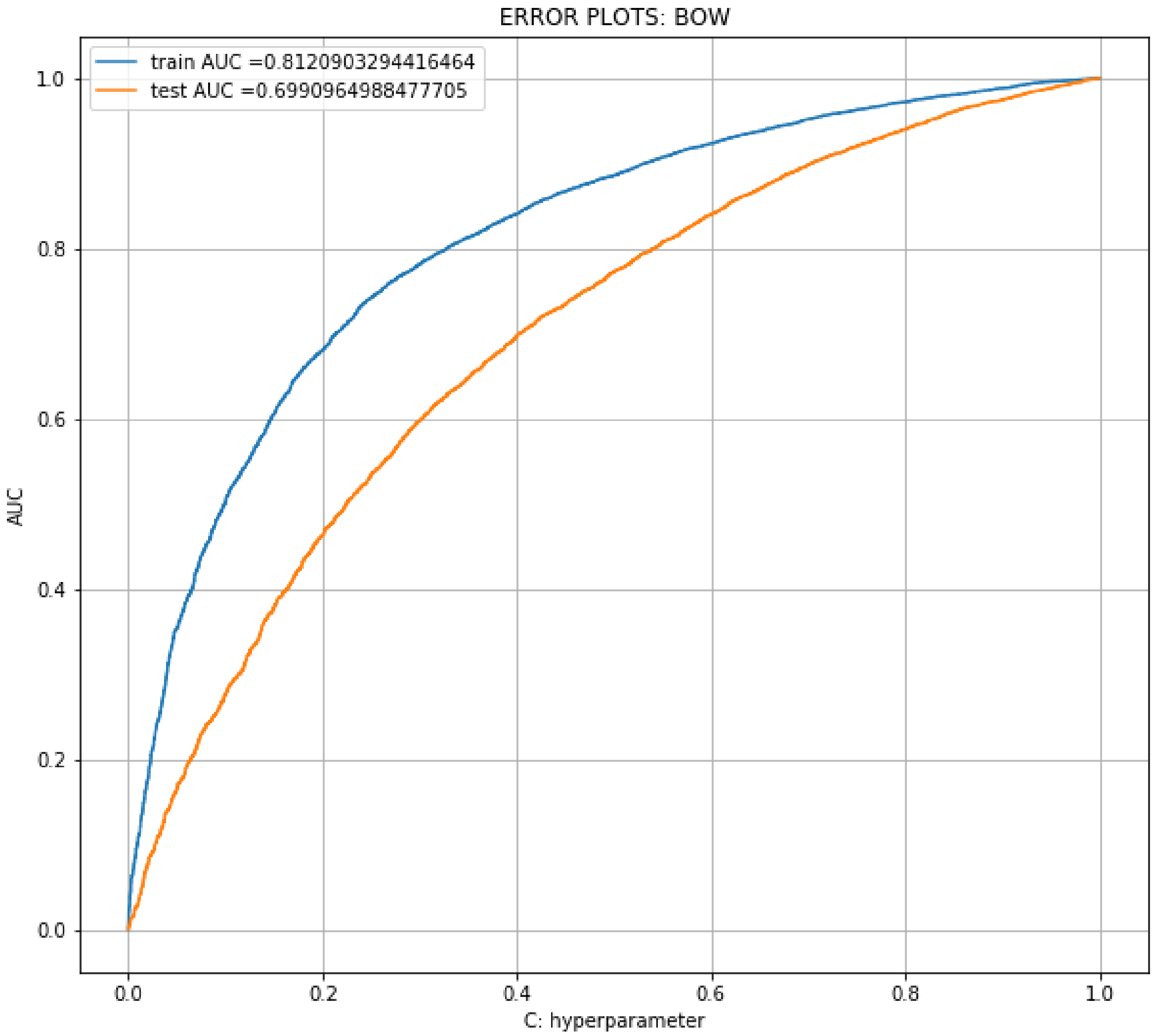
```
0.001
0.7059202526525031
```

In [0]:

```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
best_c = c_bow[cv_auc_bow.index(max(cv_auc_bow))]
c_bow = LogisticRegression(C = best_c)
c_bow.fit(X_train_bow_feat, y_train)

#predict probabilities for train and test
y_train_pred_bow = c_bow.predict_proba(X_train_bow_feat)[:,1]
y_test_pred_bow = c_bow.predict_proba(X_test_bow_feat)[:,1]

train_fpr_bow, train_tpr_bow, tr_thresholds_bow = roc_curve(y_train, y_train_pred_bow)
test_fpr_bow, test_tpr_bow, te_thresholds_bow = roc_curve(y_test, y_test_pred_bow)
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(train_fpr_bow, train_tpr_bow, label="train AUC ="+str(auc(train_fpr_bow, train_tpr
plt.plot(test_fpr_bow, test_tpr_bow, label="test AUC ="+str(auc(test_fpr_bow, test_tpr_bow)
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [0]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix = confusion_matrix(y_train, predict(y_train_pred_bow, tr_thresholds_bow, train_
print(confusion_matrix(y_train, predict(y_train_pred_bow, tr_thresholds_bow, train_fpr_bow,
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred_bow, tr_thresholds_bow, test_fpr_bow, te
```

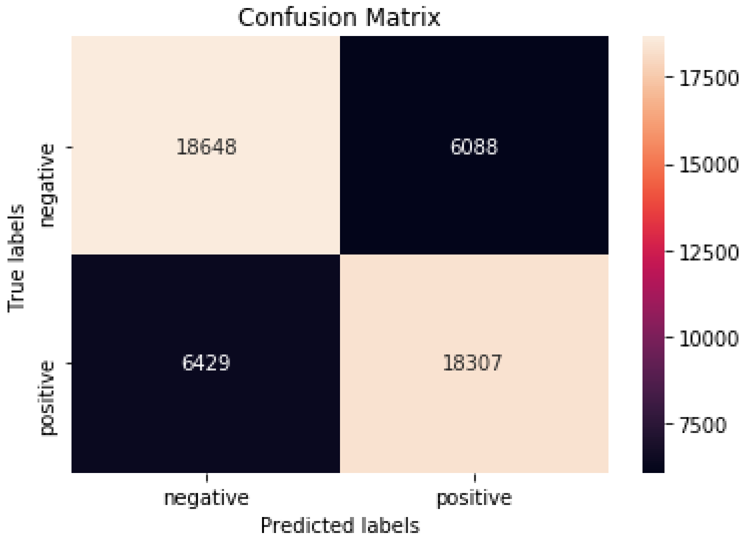
```
============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5579438534573218 for threshold 0.494
the maximum value of tpr*(1-fpr) 0.5579438534573218 for threshold 0.494
[[18648  6088]
 [ 6429 18307]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.42271370350923654 for threshold 0.509
[[ 2002  1263]
 [ 5769 12416]]
```

In [0]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[104]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



Set 2: categorical, numerical features + project_title(TFIDF) + preprocessed_eassay (TFIDF with bi-grams with min_df=10 and max_features=5000)

In [0]:

```
# Creating list of C for LR
c_tfidf = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
print(c_tfidf)
train_auc_tfidf = []
cv_auc_tfidf = []
for c in tqdm(c_tfidf):
    lr_tfidf = LogisticRegression(C=c)
    lr_tfidf.fit(X_train_tfidf_feat, y_train)

    #predict probabilities for train and validation
    y_train_pred_tfidf = lr_tfidf.predict_proba(X_train_tfidf_feat)[:,1]
    y_cv_pred_tfidf = lr_tfidf.predict_proba(X_cv_tfidf_feat)[:,1]

#     y_train_pred = batch_predict(neigh, X_tr)
#     y_cv_pred = batch_predict(neigh, X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    train_auc_tfidf.append(roc_auc_score(y_train,y_train_pred_tfidf))
    cv_auc_tfidf.append(roc_auc_score(y_cv, y_cv_pred_tfidf))
```

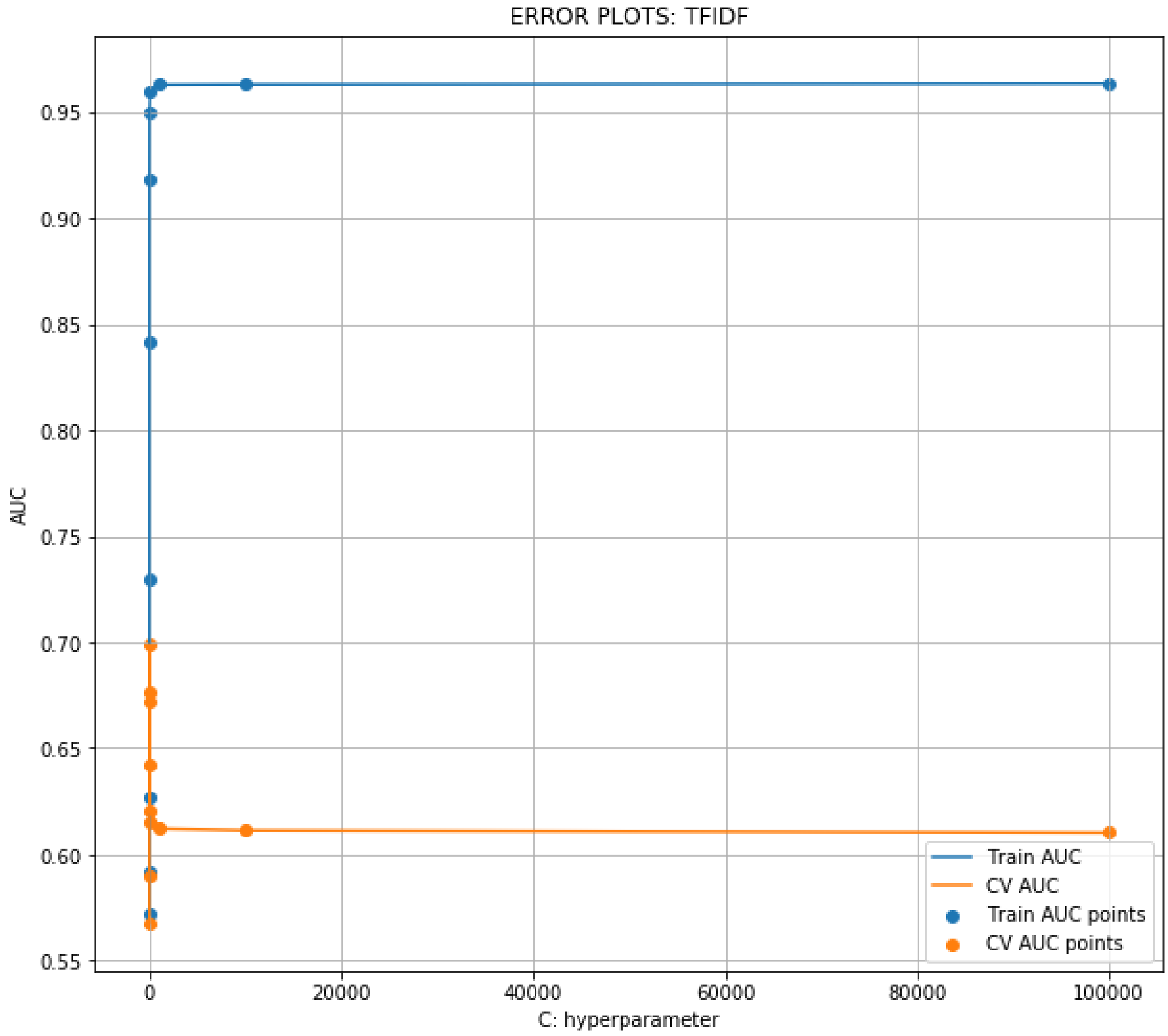
```
  0%|          | 0/11 [00:00<?, ?it/s]
```

```
[1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
```

```
100%|██████████| 11/11 [09:01<00:00, 98.56s/it]
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(c_tfidf, train_auc_tfidf, label='Train AUC')
plt.plot(c_tfidf, cv_auc_tfidf, label='CV AUC')

plt.scatter(c_tfidf, train_auc_tfidf, label='Train AUC points')
plt.scatter(c_tfidf, cv_auc_tfidf, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: TFIDF")
plt.grid()
plt.show()
```



In [0]:

```
best_c = c_tfidf[cv_auc_tfidf.index(max(cv_auc_tfidf))]
print(best_c)
print(max(cv_auc_tfidf))
```

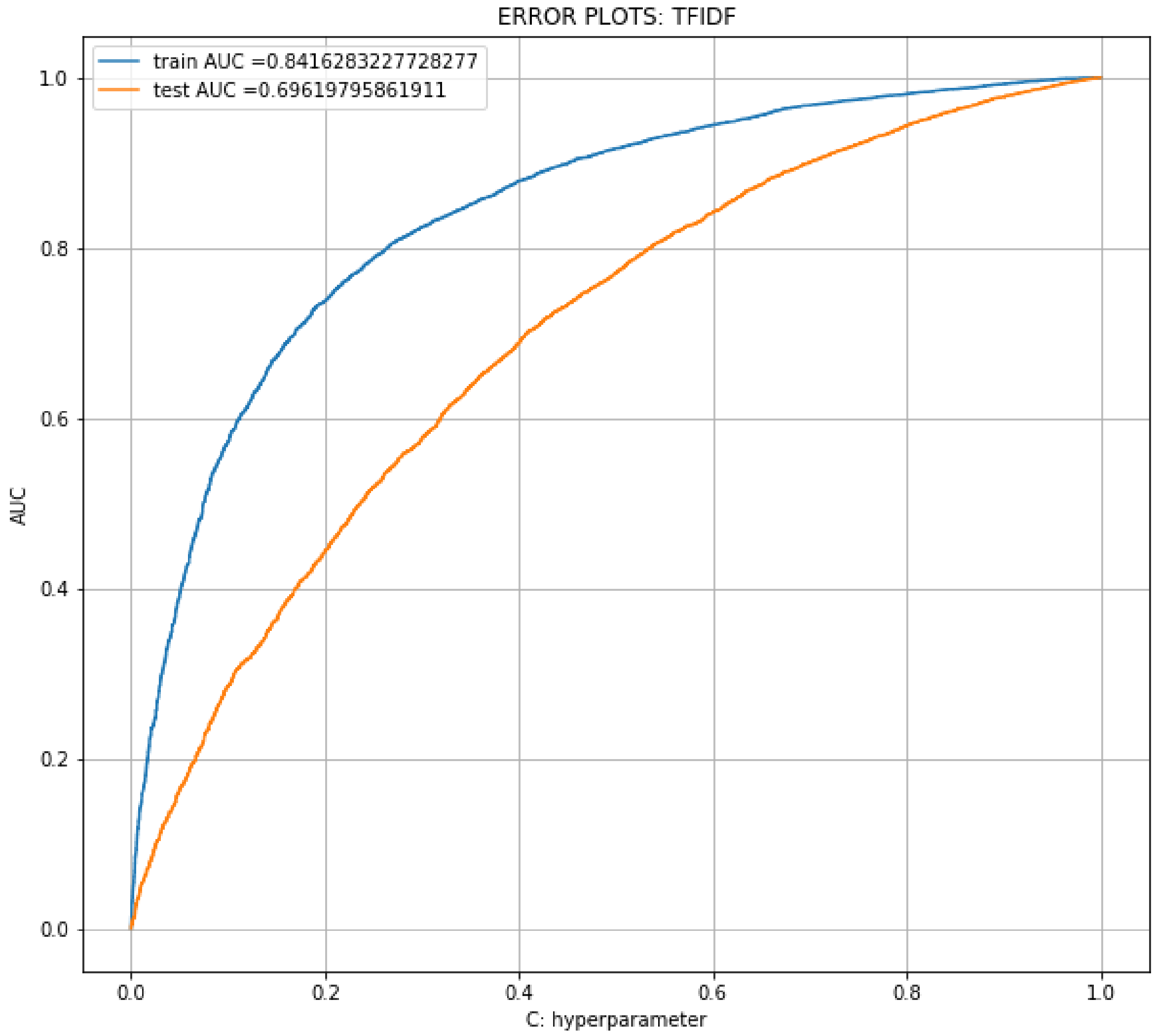
```
0.1
0.6989409268126389
```

In [0]:

```
# Training Naive Bayes with best Alpha
from sklearn.metrics import roc_curve, auc, classification_report
best_c = c_tfidf[cv_auc_tfidf.index(max(cv_auc_tfidf))]
lr_tfidf = LogisticRegression(C = best_c)
lr_tfidf.fit(X_train_tfidf_feat, y_train)

#predict probabilities for train and test
y_train_pred_tfidf = lr_tfidf.predict_proba(X_train_tfidf_feat)[:,1]
y_test_pred_tfidf = lr_tfidf.predict_proba(X_test_tfidf_feat)[:,1]

train_fpr_tfidf, train_tpr_tfidf, tr_thresholds_tfidf = roc_curve(y_train, y_train_pred_tfi
test_fpr_tfidf, test_tpr_tfidf, te_thresholds_tfidf = roc_curve(y_test, y_test_pred_tfidf)
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(train_fpr_tfidf, train_tpr_tfidf, label="train AUC ="+str(auc(train_fpr_tfidf, tra
plt.plot(test_fpr_tfidf, test_tpr_tfidf, label="test AUC ="+str(auc(test_fpr_tfidf, test_tp
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: TFIDF")
plt.grid()
plt.show()
```



In [0]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix = confusion_matrix(y_train, predict(y_train_pred_tfidf, tr_thresholds_tfidf, tr
print(confusion_matrix(y_train, predict(y_train_pred_tfidf, tr_thresholds_tfidf, train_fpr_
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred_tfidf, tr_thresholds_tfidf, test_fpr_tfi
```

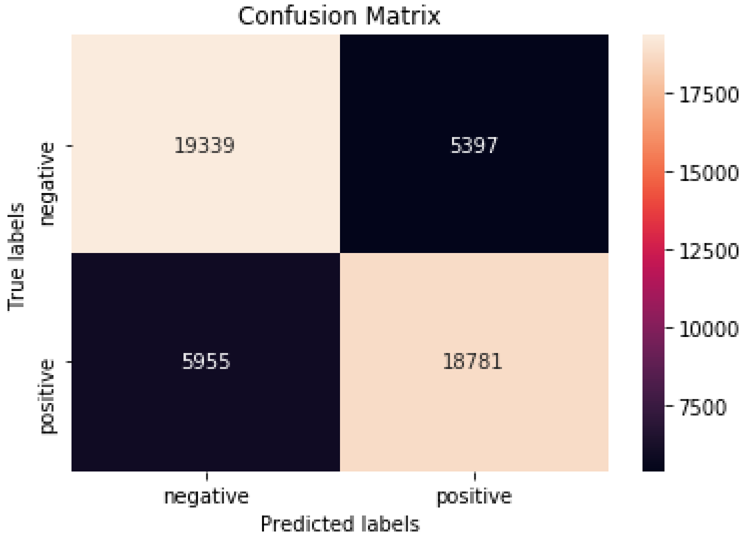
```
============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.5935998487494958 for threshold 0.493
the maximum value of tpr*(1-fpr) 0.5935998487494958 for threshold 0.493
[[19339  5397]
 [ 5955 18781]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.4160418634242836 for threshold 0.49
[[ 1799  1466]
 [ 4862 13323]]
```

In [0]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[111]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



Set 3: categorical, numerical features + project_title(Avg W2V) + preprocessed_eassay (Avg W2V)

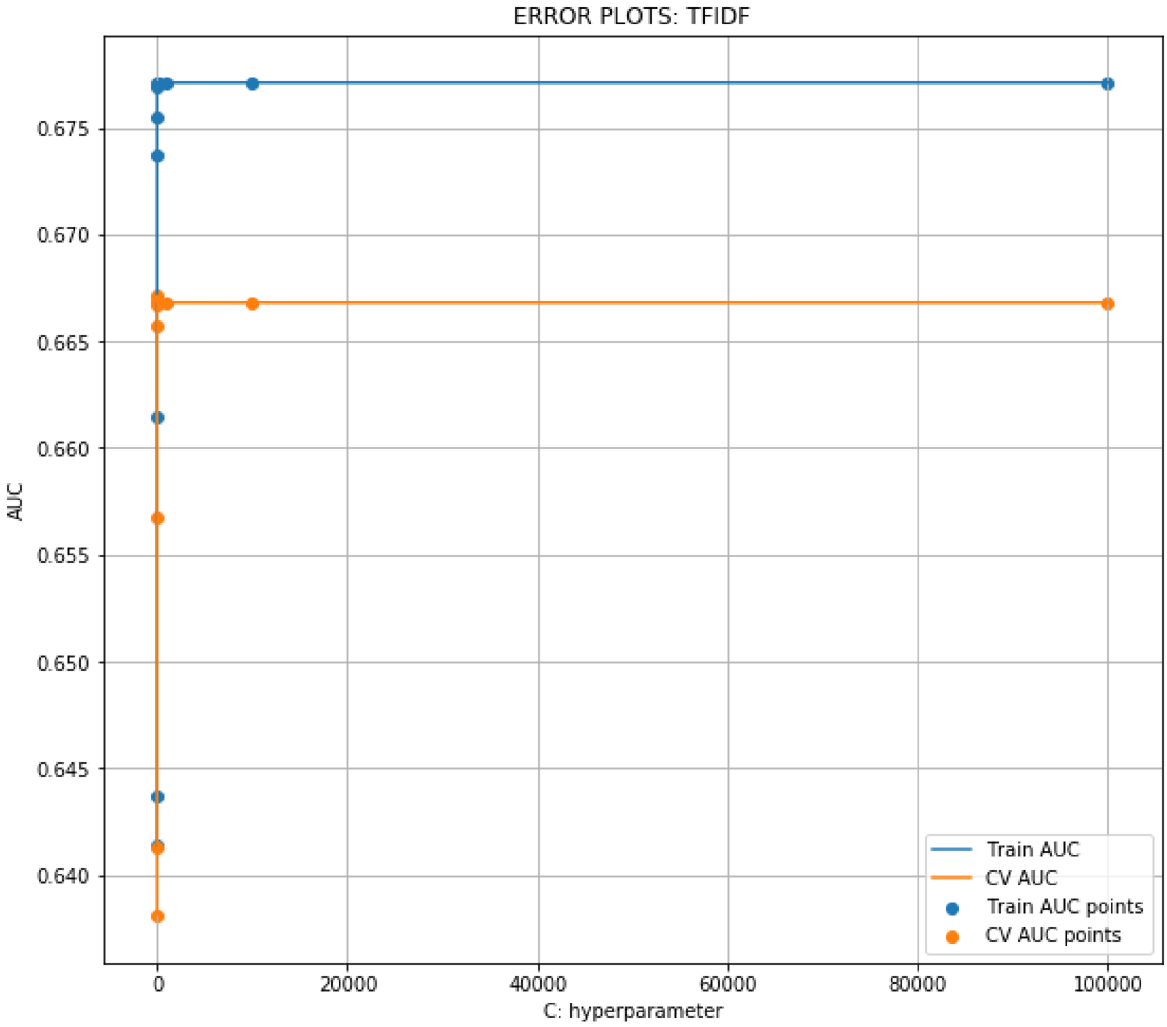
In [0]:

```
# Creating list of C for LR
c_avg_w2v = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
print(c_avg_w2v)
train_auc_avg_w2v = []
cv_auc_avg_w2v = []
for c in tqdm(c_avg_w2v):
    lr_avg_w2v = LogisticRegression(C=c)
    lr_avg_w2v.fit(X_train_avg_w2v_text, y_train)

    #predict probabilities for train and validation
    y_train_pred_avg_w2v = lr_avg_w2v.predict_proba(X_train_avg_w2v_text)[:,1]
    y_cv_pred_avg_w2v = lr_avg_w2v.predict_proba(X_cv_avg_w2v_text)[:,1]

#     y_train_pred = batch_predict(neigh, X_tr)
#     y_cv_pred = batch_predict(neigh, X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    train_auc_avg_w2v.append(roc_auc_score(y_train,y_train_pred_avg_w2v))
    cv_auc_avg_w2v.append(roc_auc_score(y_cv, y_cv_pred_avg_w2v))
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

[1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]

100%|██████████| 11/11 [00:17<00:00,  2.15s/it]
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(c_avg_w2v, train_auc_avg_w2v, label='Train AUC')
plt.plot(c_avg_w2v, cv_auc_avg_w2v, label='CV AUC')

plt.scatter(c_avg_w2v, train_auc_avg_w2v, label='Train AUC points')
plt.scatter(c_avg_w2v, cv_auc_avg_w2v, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: TFIDF")
plt.grid()
plt.show()
```



In [0]:

```
best_c = c_avg_w2v[cv_auc_avg_w2v.index(max(cv_auc_avg_w2v))]
print(best_c)
print(max(cv_auc_avg_w2v))
```

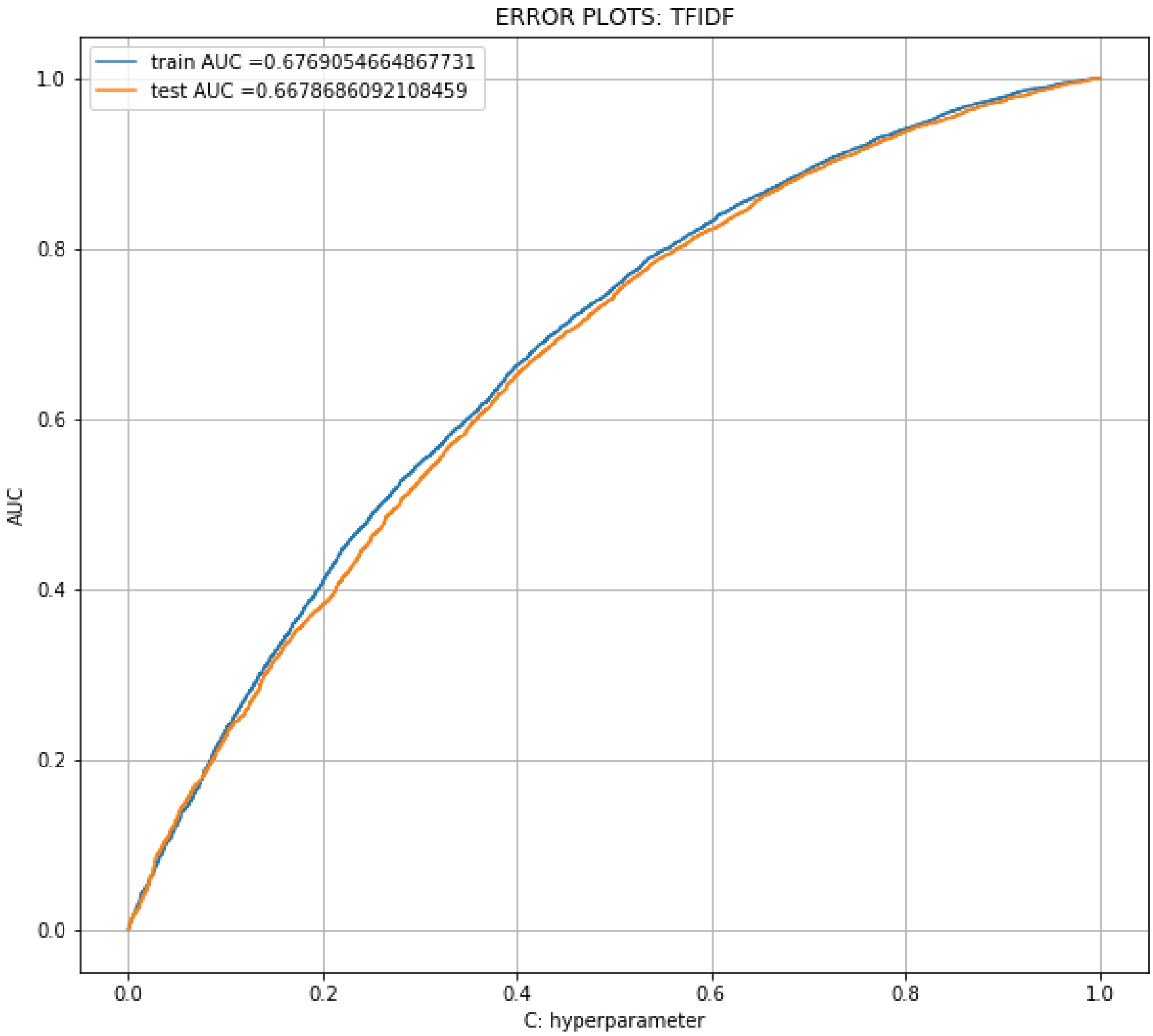
```
1
0.6671352147930393
```

In [0]:

```
# Training Naive Bayes with best Alpha
from sklearn.metrics import roc_curve, auc, classification_report
best_c = c_avg_w2v[cv_auc_avg_w2v.index(max(cv_auc_avg_w2v))]
lr_avg_w2v = LogisticRegression(C = best_c)
lr_avg_w2v.fit(X_train_avg_w2v_text, y_train)

#predict probabilities for train and test
y_train_pred_avg_w2v = lr_avg_w2v.predict_proba(X_train_avg_w2v_text)[:,1]
y_test_pred_avg_w2v = lr_avg_w2v.predict_proba(X_test_avg_w2v_text)[:,1]

train_fpr_avg_w2v, train_tpr_avg_w2v, tr_thresholds_avg_w2v = roc_curve(y_train, y_train_pr
test_fpr_avg_w2v, test_tpr_avg_w2v, te_thresholds_avg_w2v = roc_curve(y_test, y_test_pred_a
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(train_fpr_avg_w2v, train_tpr_avg_w2v, label="train AUC ="+str(auc(train_fpr_avg_w2
plt.plot(test_fpr_avg_w2v, test_tpr_avg_w2v, label="test AUC ="+str(auc(test_fpr_avg_w2v, t
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: TFIDF")
plt.grid()
plt.show()
```



In [0]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix = confusion_matrix(y_train, predict(y_train_pred_avg_w2v, tr_thresholds_avg_w2v
print(confusion_matrix(y_train, predict(y_train_pred_avg_w2v, tr_thresholds_avg_w2v, train_
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred_avg_w2v, tr_thresholds_avg_w2v, test_fpr
```

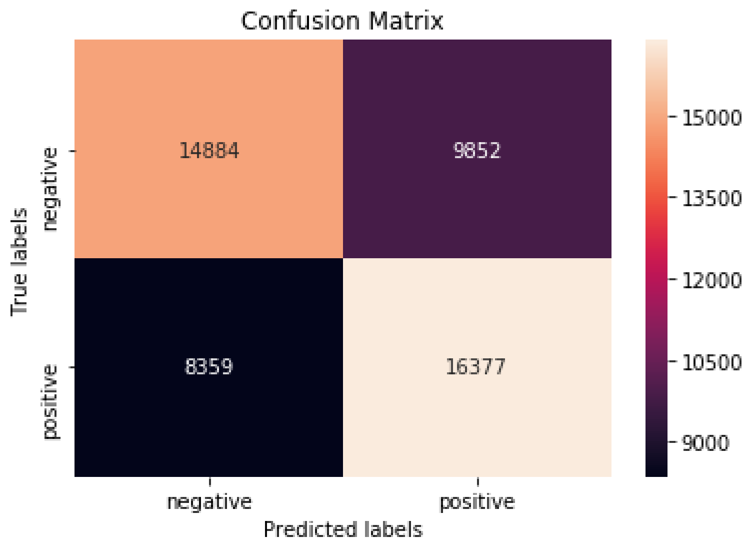
```
============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.39837774217862226 for threshold 0.484
the maximum value of tpr*(1-fpr) 0.39837774217862226 for threshold 0.484
[[14884  9852]
 [ 8359 16377]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.39148615914113954 for threshold 0.542
[[2325  940]
 [8878 9307]]
```

In [0]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[119]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



Set 4: categorical, numerical features + project_title(TFIDF W2V) + preprocessed_eassay (TFIDF W2V)

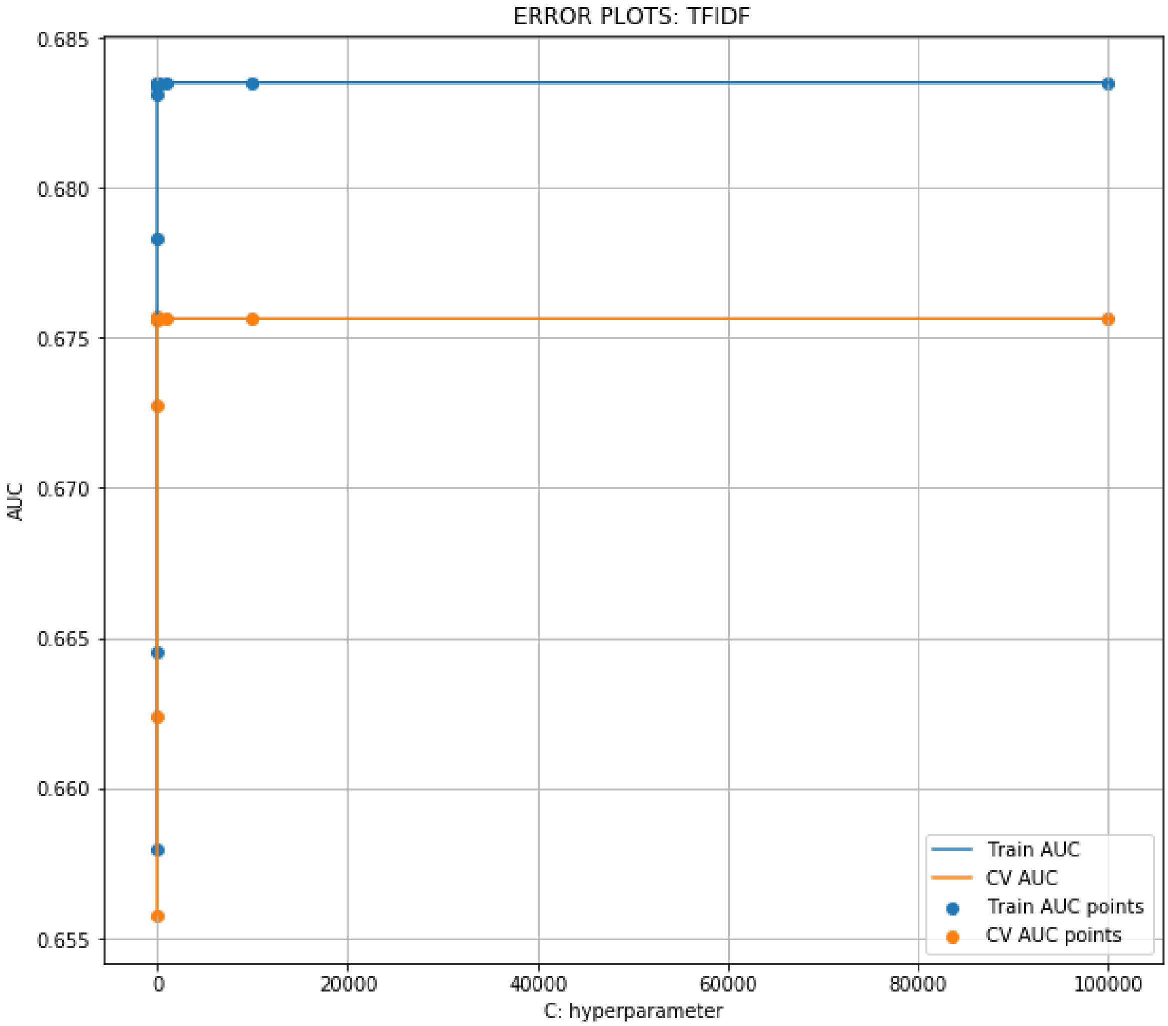
In [0]:

```
# Creating list of C for LR
c_tfidf_w2v = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
print(c_tfidf_w2v)
train_auc_tfidf_w2v = []
cv_auc_tfidf_w2v = []
for c in tqdm(c_tfidf_w2v):
    lr_tfidf_w2v = LogisticRegression(C=c)
    lr_tfidf_w2v.fit(X_train_tfidf_w2v_text, y_train)

    #predict probabilities for train and validation
    y_train_pred_tfidf_w2v = lr_tfidf_w2v.predict_proba(X_train_tfidf_w2v_text)[:,1]
    y_cv_pred_tfidf_w2v = lr_tfidf_w2v.predict_proba(X_cv_tfidf_w2v_text)[:,1]

#     y_train_pred = batch_predict(neigh, X_tr)
#     y_cv_pred = batch_predict(neigh, X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    train_auc_tfidf_w2v.append(roc_auc_score(y_train,y_train_pred_tfidf_w2v))
    cv_auc_tfidf_w2v.append(roc_auc_score(y_cv, y_cv_pred_tfidf_w2v))
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

[1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]

100%|██████████| 11/11 [00:13<00:00,  1.43s/it]
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(c_tfidf_w2v, train_auc_tfidf_w2v, label='Train AUC')
plt.plot(c_tfidf_w2v, cv_auc_tfidf_w2v, label='CV AUC')

plt.scatter(c_tfidf_w2v, train_auc_tfidf_w2v, label='Train AUC points')
plt.scatter(c_tfidf_w2v, cv_auc_tfidf_w2v, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: TFIDF")
plt.grid()
plt.show()
```



In [0]:

```
best_c = c_tfidf_w2v[cv_auc_tfidf_w2v.index(max(cv_auc_tfidf_w2v))]
print(best_c)
print(max(cv_auc_tfidf_w2v))
```

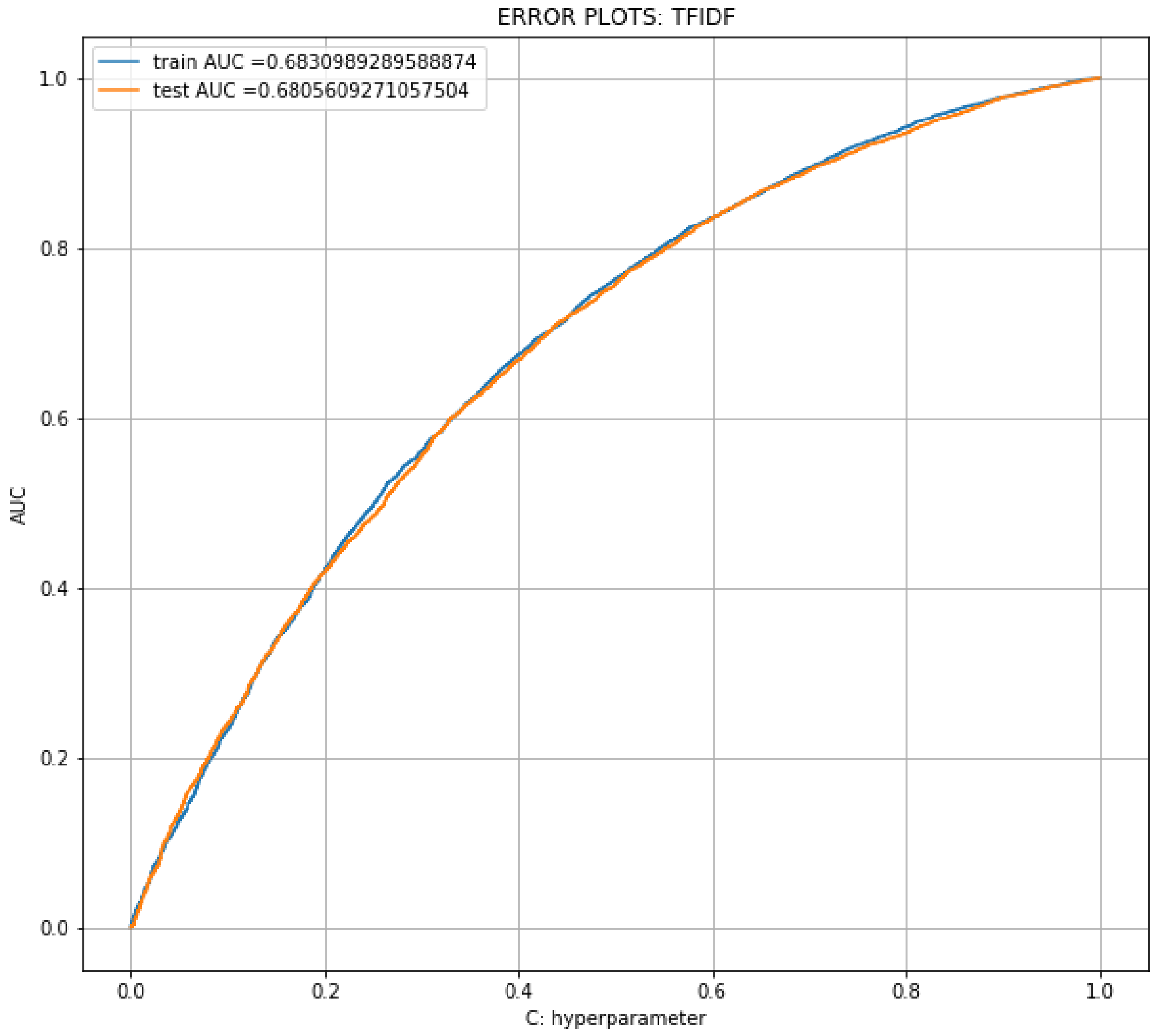
```
0.01
0.6757279416707379
```

In [0]:

```
# Training Naive Bayes with best Alpha
from sklearn.metrics import roc_curve, auc, classification_report
best_c = c_tfidf_w2v[cv_auc_tfidf_w2v.index(max(cv_auc_tfidf_w2v))]
lr_tfidf_w2v = LogisticRegression(C = best_c)
lr_tfidf_w2v.fit(X_train_tfidf_w2v_text, y_train)

#predict probabilities for train and test
y_train_pred_tfidf_w2v = lr_tfidf_w2v.predict_proba(X_train_tfidf_w2v_text)[:,1]
y_test_pred_tfidf_w2v = lr_tfidf_w2v.predict_proba(X_test_tfidf_w2v_text)[:,1]

train_fpr_tfidf_w2v, train_tpr_tfidf_w2v, tr_thresholds_tfidf_w2v = roc_curve(y_train, y_tr
test_fpr_tfidf_w2v, test_tpr_tfidf_w2v, te_thresholds_tfidf_w2v = roc_curve(y_test, y_test_
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(train_fpr_tfidf_w2v, train_tpr_tfidf_w2v, label="train AUC ="+str(auc(train_fpr_tf
plt.plot(test_fpr_tfidf_w2v, test_tpr_tfidf_w2v, label="test AUC ="+str(auc(test_fpr_tfidf_
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: TFIDF")
plt.grid()
plt.show()
```



In [0]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix = confusion_matrix(y_train, predict(y_train_pred_tfidf_w2v, tr_thresholds_tfidf
print(confusion_matrix(y_train, predict(y_train_pred_tfidf_w2v, tr_thresholds_tfidf_w2v, tr
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred_tfidf_w2v, tr_thresholds_tfidf_w2v, test
```

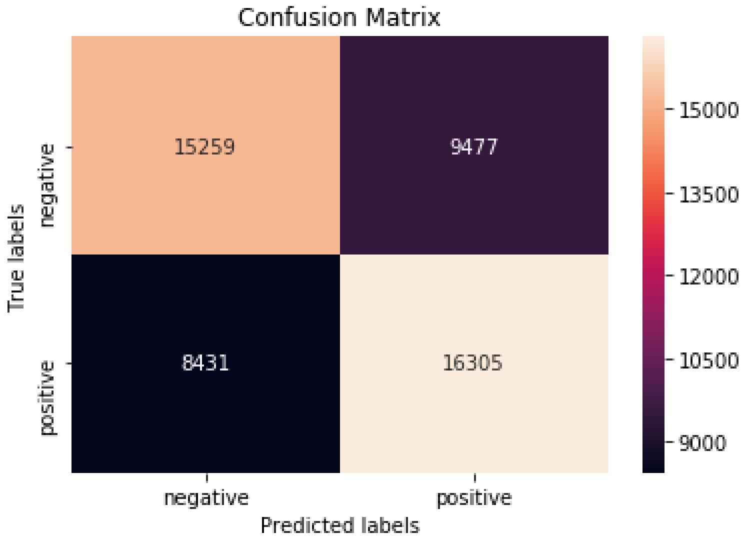
```
============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.4066192469188734 for threshold 0.481
the maximum value of tpr*(1-fpr) 0.4066192469188734 for threshold 0.481
[[15259  9477]
 [ 8431 16305]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.40260683017531657 for threshold 0.534
[[2396  869]
 [8901 9284]]
```

In [0]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[128]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



2.5 Logistic Regression with added Features Set 5

In [0]:

```
# we get the cost of the project using resource.csv file
resource_data.head(2)
```

Out[46]:

	id	description	quantity	price
0	p233245	LC652 - Lakeshore Double-Space Mobile Drying Rack	1	149.00
1	p069063	Bouncy Bands for Desks (Blue support pipes)	3	14.95

In [0]:

```
# https://stackoverflow.com/questions/22407798/how-to-reset-a-dataframes-indexes-for-all-gr
price_data = resource_data.groupby('id').agg({'price':'sum', 'quantity':'sum'}).reset_index
price_data.head(2)
```

Out[47]:

	id	price	quantity
0	p000001	459.56	7
1	p000002	515.89	21

In [0]:

```
# join two dataframes in python:
cleaned_project_data = pd.merge(cleaned_project_data, price_data, on='id', how='left')
```

In [0]:

```
cleaned_project_data.head(2)
```

Out[49]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategori
0	p253737	my students english learners working english s...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

In [0]:

```
cleaned_project_data['words_in_project_title'] = cleaned_project_data.apply(lambda row: len
cleaned_project_data['words_in_essays'] = cleaned_project_data.apply(lambda row: len(row.cl
```

In [0]:

```
import nltk
from nltk.sentiment.vader import SentimentIntensityAnalyzer

sid = SentimentIntensityAnalyzer()
negative = []
positive = []
neutral = []
compound = []
for text in tqdm(cleaned_project_data['cleaned_essay']):
  ss = sid.polarity_scores(text)
  negative.append(ss['neg'])
  positive.append(ss['pos'])
  neutral.append(ss['neu'])
  compound.append(ss['compound'])
```

```
100%|██████████| 109248/109248 [04:08<00:00, 439.12it/s]
```

In [0]:

```
cleaned_project_data['negative'] = negative
cleaned_project_data['positive'] = positive
cleaned_project_data['neutral'] = neutral
cleaned_project_data['compound'] = compound
```

In [0]:

```
# Taking complete data
project_65k_new = cleaned_project_data[:65000].copy()
project_65k_new.shape
```

Out[100]:

```
(65000, 19)
```

In [0]:

```
set(project_65k_new['teacher_prefix'])
```

Out[101]:

```
{'dr', 'mr', 'mrs', 'ms', 'nan', 'teacher'}
```

In [0]:

```
print(len(project_65k_new[project_65k_new['project_is_approved'] == 0]))
print(len(project_65k_new[project_65k_new['project_is_approved'] == 1]))
```

```
9895
55105
```

In [0]:

```
X = project_65k_new.drop(['project_is_approved'], axis=1)
y = project_65k_new['project_is_approved']
print(type(y))
```

```
<class 'pandas.core.series.Series'>
```

In [0]:

```
# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, stratify=y)
X_train, X_cv, y_train, y_cv = train_test_split(X_train, y_train, test_size=0.33, stratify=
```

In [0]:

```
print(set(X_train['teacher_prefix'].values))
print(set(X_cv['teacher_prefix'].values))
print(set(X_test['teacher_prefix'].values))
```

```
{'mr', 'ms', 'nan', 'teacher', 'mrs', 'dr'}
{'mr', 'ms', 'nan', 'teacher', 'mrs', 'dr'}
{'mr', 'ms', 'teacher', 'mrs', 'dr'}
```

In [0]:

```
print(y_train.value_counts())
print(len(y_train == 0))
print(len(y_train == 1))
```

```
1    24736
0     4442
Name: project_is_approved, dtype: int64
29178
29178
```

In [0]:

```
test = pd.concat([X_train, y_train], axis=1)
test.head(2)
print(len(test[test['project_is_approved'] == 0]))
print(len(test[test['project_is_approved'] == 1]))
```

```
4442
24736
```

In [0]:

```
from sklearn.utils import resample
```

In [0]:

```
#https://elitedatascience.com/imbalanced-classes
# Separate majority and minority classes
project_majority = test[test.project_is_approved==1]
project_minority = test[test.project_is_approved==0]

# Upsample minority class
project_minority_upsampled = resample(project_minority,
                                  replace=True,     # sample with replacement
                                  n_samples=24736,    # to match majority class
                                  random_state=123) # reproducible results

# Combine majority class with upsampled minority class
project_upsampled = pd.concat([project_majority, project_minority_upsampled])

# Display new class counts
project_upsampled.project_is_approved.value_counts()
```

Out[109]:

```
1    24736
0    24736
Name: project_is_approved, dtype: int64
```

In [0]:

```
X_train = project_upsampled.drop(['project_is_approved'], axis=1)
y_train = project_upsampled['project_is_approved']
```

In [0]:

```
print(y_train.value_counts())
```

```
1    24736
0    24736
Name: project_is_approved, dtype: int64
```

In [0]:

```
print(len(X_train), len(y_train))
print(len(X_cv), len(y_cv))
print(len(X_test), len(y_test))
```

```
49472 49472
14372 14372
21450 21450
```

In [0]:

```
print(set(X_train['teacher_prefix'].values))
print(set(X_cv['teacher_prefix'].values))
print(set(X_test['teacher_prefix'].values))
```

```
{'mr', 'ms', 'nan', 'teacher', 'mrs', 'dr'}
{'mr', 'ms', 'nan', 'teacher', 'mrs', 'dr'}
{'mr', 'ms', 'teacher', 'mrs', 'dr'}
```

In [0]:

```
cleaned_project_data.head(2)
```

Out[114]:

	id	cleaned_essay	cleaned_project_title	clean_categories	clean_subcategori
0	p253737	my students english learners working english s...	educational support english learners home	Literacy_Language	ESL Literacy
1	p258326	our students arrive school eager learn they po...	wanted projector hungry learners	History_Civics Health_Sports	Civics_Government TeamSports

Project Category

In [0]:

```
# we use count vectorizer to convert the values into one hot encoded features
# Project Category
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(vocabulary=list(sorted_cat_dict.keys()), lowercase=False, bina
X_train_one_hot_clean_cat = vectorizer.fit_transform(X_train['clean_categories'].values)
X_cv_one_hot_clean_cat = vectorizer.transform(X_cv['clean_categories'].values)
X_test_one_hot_clean_cat = vectorizer.transform(X_test['clean_categories'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_cat.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_cat.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_cat.shape)
```

```
['Warmth', 'Care_Hunger', 'History_Civics', 'Music_Arts', 'AppliedLearning',
'SpecialNeeds', 'Health_Sports', 'Math_Science', 'Literacy_Language']
Shape of matrix after one hot encodig  (49472, 9)
Shape of matrix after one hot encodig  (14372, 9)
Shape of matrix after one hot encodig  (21450, 9)
```

Project Sub-category

In [0]:

```
# Project Sub-category
vectorizer = CountVectorizer(vocabulary=list(sorted_sub_cat_dict.keys()), lowercase=False,
X_train_one_hot_clean_sub_cat = vectorizer.fit_transform(X_train['clean_subcategories'].val
X_cv_one_hot_clean_sub_cat = vectorizer.transform(X_cv['clean_subcategories'].values)
X_test_one_hot_clean_sub_cat = vectorizer.transform(X_test['clean_subcategories'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_sub_cat.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_sub_cat.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_sub_cat.shape)
```

```
['Economics', 'CommunityService', 'FinancialLiteracy', 'ParentInvolvement',
'Extracurricular', 'Civics_Government', 'ForeignLanguages', 'NutritionEducat
ion', 'Warmth', 'Care_Hunger', 'SocialSciences', 'PerformingArts', 'Characte
rEducation', 'TeamSports', 'Other', 'College_CareerPrep', 'Music', 'History_
Geography', 'Health_LifeScience', 'EarlyDevelopment', 'ESL', 'Gym_Fitness',
'EnvironmentalScience', 'VisualArts', 'Health_Wellness', 'AppliedSciences',
'SpecialNeeds', 'Literature_Writing', 'Mathematics', 'Literacy']
Shape of matrix after one hot encodig  (49472, 30)
Shape of matrix after one hot encodig  (14372, 30)
Shape of matrix after one hot encodig  (21450, 30)
```

School State

In [0]:

```
# School State
vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_school_state = vectorizer.fit_transform(X_train['school_state'].value
X_cv_one_hot_clean_school_state = vectorizer.transform(X_cv['school_state'].values)
X_test_one_hot_clean_school_state = vectorizer.transform(X_test['school_state'].values)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_school_state.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_school_state.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_school_state.shape)
```

```
['AK', 'AL', 'AR', 'AZ', 'CA', 'CO', 'CT', 'DC', 'DE', 'FL', 'GA', 'HI', 'I
A', 'ID', 'IL', 'IN', 'KS', 'KY', 'LA', 'MA', 'MD', 'ME', 'MI', 'MN', 'MO',
'MS', 'MT', 'NC', 'ND', 'NE', 'NH', 'NJ', 'NM', 'NV', 'NY', 'OH', 'OK', 'O
R', 'PA', 'RI', 'SC', 'SD', 'TN', 'TX', 'UT', 'VA', 'VT', 'WA', 'WI', 'WV',
'WY']
Shape of matrix after one hot encodig  (49472, 51)
Shape of matrix after one hot encodig  (14372, 51)
Shape of matrix after one hot encodig  (21450, 51)
```

Teacher Prefix

```
In [0]:
```

```
# Teacher Prefix
import re
X_train_prefix_list = []
X_cv_prefix_list = []
X_test_prefix_list = []
for s in tqdm(X_train['teacher_prefix'].values):
    train_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    train_prefix = re.sub('nan', '', str(train_prefix))
    X_train_prefix_list.append(train_prefix)
for s in tqdm(X_cv['teacher_prefix'].values):
    cv_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    X_cv_prefix_list.append(cv_prefix)
for s in tqdm(X_test['teacher_prefix'].values):
    test_prefix = re.sub('[^A-Za-z0-9]+', '', str(s))
    test_prefix = re.sub('nan', '', str(test_prefix))
    X_test_prefix_list.append(test_prefix)

vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_teacher_prefix = vectorizer.fit_transform(X_train_prefix_list)
X_cv_one_hot_clean_teacher_prefix = vectorizer.transform(X_cv_prefix_list)
X_test_one_hot_clean_teacher_prefix = vectorizer.fit_transform(X_test_prefix_list)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_teacher_prefix.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_teacher_prefix.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_teacher_prefix.shape)
```

```
100%|██████████| 49472/49472 [00:00<00:00, 325805.97it/s]
100%|██████████| 14372/14372 [00:00<00:00, 492395.52it/s]
100%|██████████| 21450/21450 [00:00<00:00, 324716.21it/s]

['dr', 'mr', 'mrs', 'ms', 'teacher']
Shape of matrix after one hot encodig  (49472, 5)
Shape of matrix after one hot encodig  (14372, 5)
Shape of matrix after one hot encodig  (21450, 5)
```

Project Grade Category

```
In [0]:
```

```
# Project Grade Category

def grade_cat_cleaning(data):
  proj_grade_cat_list = []
  for grade in tqdm(data):
#       grade_cat = re.sub('-',' to ', grade)
#       grade_cat = re.sub('2',' two ', grade_cat)
#       grade_cat = re.sub('3',' three ', grade_cat)
#       grade_cat = re.sub('5',' five ', grade_cat)
#       grade_cat = re.sub('6',' six ', grade_cat)
#       grade_cat = re.sub('8',' eight ', grade_cat)
#       grade_cat = re.sub('9',' nine ', grade_cat)
#       grade_cat = re.sub('12',' twelve ', grade_cat)
      proj_grade_cat_list.append(grade.lower().strip())
  return proj_grade_cat_list
X_train_proj_grade_cat = grade_cat_cleaning([sent for sent in X_train['project_grade_catego
X_cv_proj_grade_cat = grade_cat_cleaning([sent for sent in X_cv['project_grade_category'].v
X_test_proj_grade_cat = grade_cat_cleaning([sent for sent in X_test['project_grade_category
vectorizer = CountVectorizer(lowercase=False, binary=True)
X_train_one_hot_clean_project_grade = vectorizer.fit_transform(X_train_proj_grade_cat)
X_cv_one_hot_clean_project_grade = vectorizer.transform(X_cv_proj_grade_cat)
X_test_one_hot_clean_project_grade = vectorizer.transform(X_test_proj_grade_cat)
print(vectorizer.get_feature_names())

print("Shape of matrix after one hot encodig ",X_train_one_hot_clean_project_grade.shape)
print("Shape of matrix after one hot encodig ",X_cv_one_hot_clean_project_grade.shape)
print("Shape of matrix after one hot encodig ",X_test_one_hot_clean_project_grade.shape)
```

```
100%|██████████| 49472/49472 [00:00<00:00, 1287751.86it/s]
100%|██████████| 14372/14372 [00:00<00:00, 1440670.55it/s]
100%|██████████| 21450/21450 [00:00<00:00, 1368019.78it/s]

['grades_3_5', 'grades_6_8', 'grades_9_12', 'grades_prek_2']
Shape of matrix after one hot encodig  (49472, 4)
Shape of matrix after one hot encodig  (14372, 4)
Shape of matrix after one hot encodig  (21450, 4)
```

teacher_number_of_previously_posted_projects

In [0]:

```
from scipy.sparse import csr_matrix
X_train_no_of_projects = np.array(X_train['teacher_number_of_previously_posted_projects'])
X_train_no_of_projects = csr_matrix(X_train_no_of_projects).T
X_cv_no_of_projects = np.array(X_cv['teacher_number_of_previously_posted_projects'])
X_cv_no_of_projects = csr_matrix(X_cv_no_of_projects).T
X_test_no_of_projects = np.array(X_test['teacher_number_of_previously_posted_projects'])
X_test_no_of_projects = csr_matrix(X_test_no_of_projects).T
print(X_train_no_of_projects.shape)
print(X_cv_no_of_projects.shape)
print(X_test_no_of_projects.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

price

In [0]:

```
from scipy.sparse import csr_matrix
X_train_price = np.array(X_train['price'])
X_train_price = csr_matrix(X_train_price).T
X_cv_price = np.array(X_cv['price'])
X_cv_price = csr_matrix(X_cv_price).T
X_test_price = np.array(X_test['price'])
X_test_price = csr_matrix(X_test_price).T
print(X_train_price.shape)
print(X_cv_price.shape)
print(X_test_price.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

quantity

In [0]:

```
X_train_quantity = np.array(X_train['quantity'])
X_train_quantity = csr_matrix(X_train_quantity).T
X_cv_quantity = np.array(X_cv['quantity'])
X_cv_quantity = csr_matrix(X_cv_quantity).T
X_test_quantity = np.array(X_test['quantity'])
X_test_quantity = csr_matrix(X_test_quantity).T
print(X_train_quantity.shape)
print(X_cv_quantity.shape)
print(X_test_quantity.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

words_in_project_title

In [0]:

```
X_train_words_in_project_title = np.array(X_train['words_in_project_title'])
X_train_words_in_project_title = csr_matrix(X_train_words_in_project_title).T
X_cv_words_in_project_title = np.array(X_cv['words_in_project_title'])
X_cv_words_in_project_title = csr_matrix(X_cv_words_in_project_title).T
X_test_words_in_project_title = np.array(X_test['words_in_project_title'])
X_test_words_in_project_title = csr_matrix(X_test_words_in_project_title).T
print(X_train_words_in_project_title.shape)
print(X_cv_words_in_project_title.shape)
print(X_test_words_in_project_title.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

words_in_essays

In [0]:

```
X_train_words_in_essays = np.array(X_train['words_in_essays'])
X_train_words_in_essays = csr_matrix(X_train_words_in_essays).T
X_cv_words_in_essays = np.array(X_cv['words_in_essays'])
X_cv_words_in_essays = csr_matrix(X_cv_words_in_essays).T
X_test_words_in_essays = np.array(X_test['words_in_essays'])
X_test_words_in_essays = csr_matrix(X_test_words_in_essays).T
print(X_train_words_in_essays.shape)
print(X_cv_words_in_essays.shape)
print(X_test_words_in_essays.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

Negative Sentiment: Essay

In [0]:

```
X_train_negative = np.array(X_train['negative'])
X_train_negative = csr_matrix(X_train_negative).T
X_cv_negative = np.array(X_cv['negative'])
X_cv_negative = csr_matrix(X_cv_negative).T
X_test_negative = np.array(X_test['negative'])
X_test_negative = csr_matrix(X_test_negative).T
print(X_train_negative.shape)
print(X_cv_negative.shape)
print(X_test_negative.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

Positive Sentiment: Essay

In [0]:

```
X_train_positive = np.array(X_train['positive'])
X_train_positive = csr_matrix(X_train_positive).T
X_cv_positive = np.array(X_cv['positive'])
X_cv_positive = csr_matrix(X_cv_positive).T
X_test_positive = np.array(X_test['positive'])
X_test_positive = csr_matrix(X_test_positive).T
print(X_train_positive.shape)
print(X_cv_positive.shape)
print(X_test_positive.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

Neutral Sentiment: Essay

In [0]:

```
import numpy as np
X_train_neutral = np.array(X_train['neutral'])
X_train_neutral = csr_matrix(X_train_neutral).T
X_cv_neutral = np.array(X_cv['neutral'])
X_cv_neutral = csr_matrix(X_cv_neutral).T
X_test_neutral = np.array(X_test['neutral'])
X_test_neutral = csr_matrix(X_test_neutral).T
print(X_train_neutral.shape)
print(X_cv_neutral.shape)
print(X_test_neutral.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

Compound Sentiment: Essay

In [0]:

```
import numpy as np
X_train_compound = np.array(X_train['compound'])
X_train_compound = csr_matrix(X_train_compound).T
X_cv_compound = np.array(X_cv['compound'])
X_cv_compound = csr_matrix(X_cv_compound).T
X_test_compound = np.array(X_test['compound'])
X_test_compound = csr_matrix(X_test_compound).T
print(X_train_compound.shape)
print(X_cv_compound.shape)
print(X_test_compound.shape)
```

```
(49472, 1)
(14372, 1)
(21450, 1)
```

Merging categorical and numerical features

In [0]:

```
from scipy.sparse import coo_matrix, hstack
print(X_train_one_hot_clean_cat.shape, X_train_one_hot_clean_sub_cat.shape, X_train_one_hot
                                 X_train_one_hot_clean_teacher_prefix.shape, X_train_one_ho
X_train_categorical_numerical = hstack([X_train_one_hot_clean_cat, X_train_one_hot_clean_su
                              X_train_one_hot_clean_teacher_prefix, X_train_one_hot_clean_p
                              X_train_price, X_train_quantity, X_train_words_in_project_tit
                              X_train_negative, X_train_positive, X_train_neutral, X_train_
X_cv_categorical_numerical = hstack([X_cv_one_hot_clean_cat, X_cv_one_hot_clean_sub_cat, X_
                                X_cv_one_hot_clean_teacher_prefix, X_cv_one_hot_clean_proje
                                 X_cv_price, X_cv_quantity, X_cv_words_in_project_title, X_
                                  X_cv_negative, X_cv_positive, X_cv_neutral, X_cv_compound
X_test_categorical_numerical = hstack([X_test_one_hot_clean_cat, X_test_one_hot_clean_sub_c
                                X_test_one_hot_clean_teacher_prefix, X_test_one_hot_clean_p
                                X_test_price, X_test_quantity, X_test_words_in_project_titl
                                 X_test_negative, X_test_positive, X_test_neutral, X_test_c
print(X_train_categorical_numerical.shape, X_cv_categorical_numerical.shape, X_test_categor
```

```
(49472, 9) (49472, 30) (49472, 51) (49472, 5) (49472, 4)
(49472, 108) (14372, 108) (21450, 108)
```

In [0]:

```
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_auc_score
```

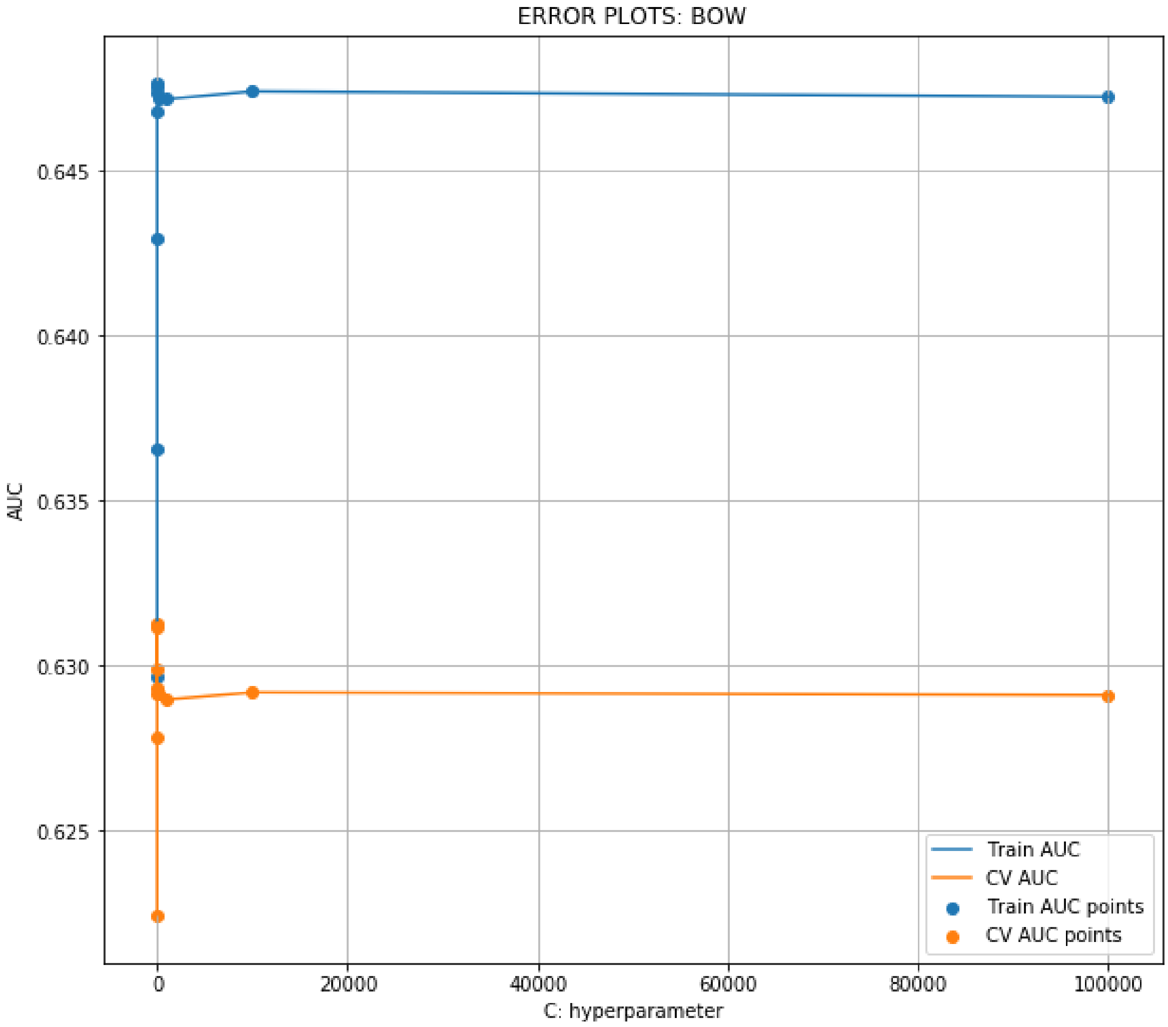
In [0]:

```
# Creating list of C for LR
c_list = [0.00001, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]
print(c_list)
train_auc = []
cv_auc = []
for c in tqdm(c_list):
    lr = LogisticRegression(C=c)
    lr.fit(X_train_categorical_numerical, y_train)

    #predict probabilities for train and validation
    y_train_pred = lr.predict_proba(X_train_categorical_numerical)[:,1]
    y_cv_pred = lr.predict_proba(X_cv_categorical_numerical)[:,1]

#     y_train_pred = batch_predict(neigh, X_tr)
#     y_cv_pred = batch_predict(neigh, X_cr)

    # roc_auc_score(y_true, y_score) the 2nd parameter should be probability estimates of t
    # not the predicted outputs
    train_auc.append(roc_auc_score(y_train,y_train_pred))
    cv_auc.append(roc_auc_score(y_cv, y_cv_pred))
```

```
  0%|          | 0/11 [00:00<?, ?it/s]

[1e-05, 0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000, 100000]

100%|██████████| 11/11 [00:11<00:00,  1.18s/it]
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(c_list, train_auc, label='Train AUC')
plt.plot(c_list, cv_auc, label='CV AUC')

plt.scatter(c_list, train_auc, label='Train AUC points')
plt.scatter(c_list, cv_auc, label='CV AUC points')

plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
best_c = c_list[cv_auc.index(max(cv_auc))]
print(best_c)
print(max(cv_auc))
```

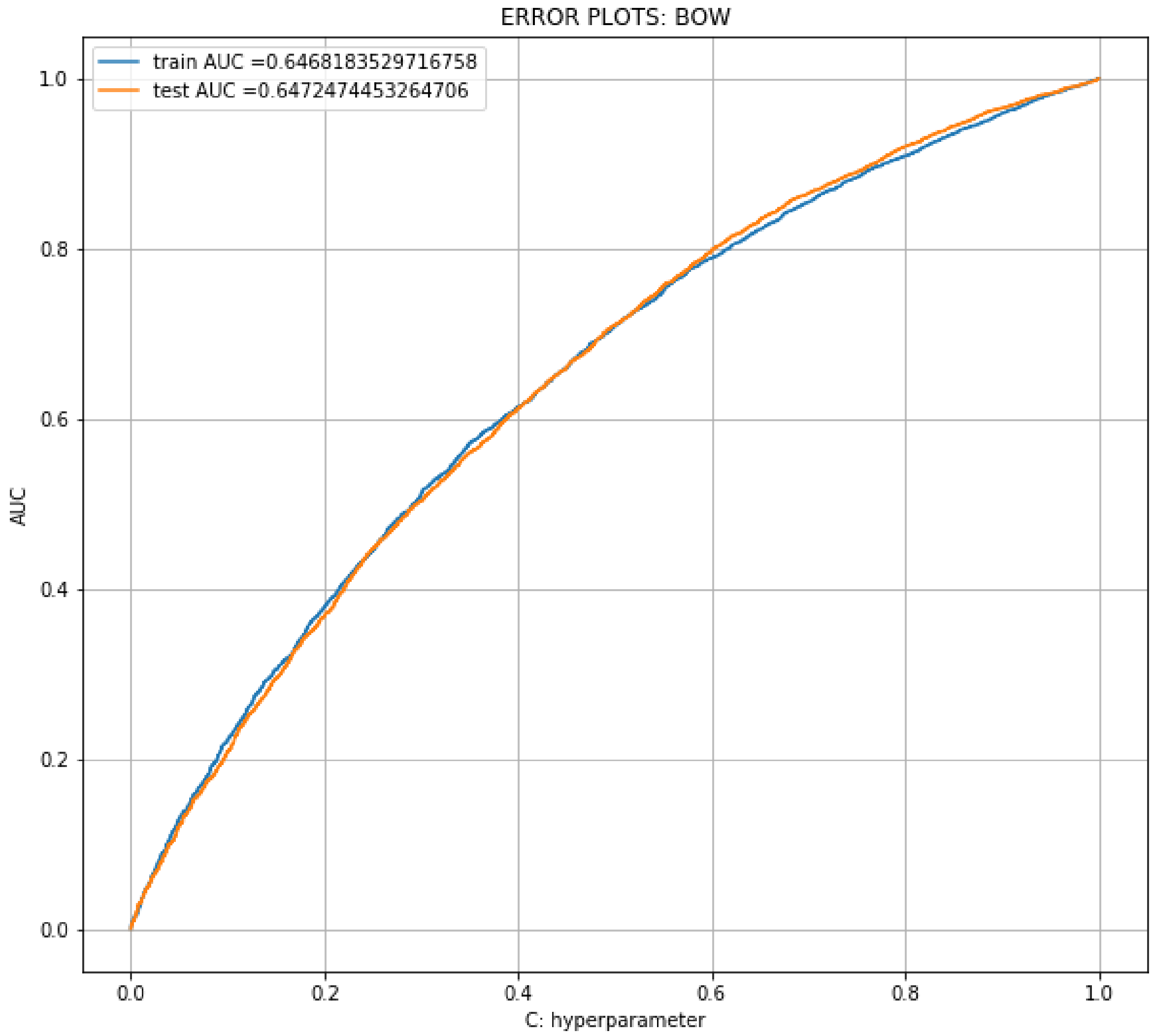
```
0.01
0.6313046465469745
```

In [0]:

```
# Training Naive Bayes with best alpha
from sklearn.metrics import roc_curve, auc, classification_report
best_c = c_list[cv_auc.index(max(cv_auc))]
lr = LogisticRegression(C = best_c)
lr.fit(X_train_categorical_numerical, y_train)

#predict probabilities for train and test
y_train_pred = lr.predict_proba(X_train_categorical_numerical)[:,1]
y_test_pred = lr.predict_proba(X_test_categorical_numerical)[:,1]

train_fpr, train_tpr, tr_thresholds = roc_curve(y_train, y_train_pred)
test_fpr, test_tpr, te_thresholds = roc_curve(y_test, y_test_pred)
```

In [0]:

```
plt.figure(figsize = (10,9))
plt.plot(train_fpr, train_tpr, label="train AUC ="+str(auc(train_fpr, train_tpr)))
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.legend()
plt.xlabel("C: hyperparameter")
plt.ylabel("AUC")
plt.title("ERROR PLOTS: BOW")
plt.grid()
plt.show()
```



In [0]:

```
# we are writing our own function for predict, with defined thresould
# we will pick a threshold that will give the least fpr
def predict(proba, threshould, fpr, tpr):

    t = threshould[np.argmax(tpr*(1-fpr))]

    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very high

    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for threshold", np.round(t
    predictions = []
    for i in proba:
        if i>=t:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions
```

In [0]:

```
print("="*100)
from sklearn.metrics import confusion_matrix, classification_report
print("Train confusion matrix")
conf_matrix = confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, tra
print(confusion_matrix(y_train, predict(y_train_pred, tr_thresholds, train_fpr, train_tpr))
print("Test confusion matrix")
print(confusion_matrix(y_test, predict(y_test_pred, tr_thresholds, test_fpr, test_tpr)))
```

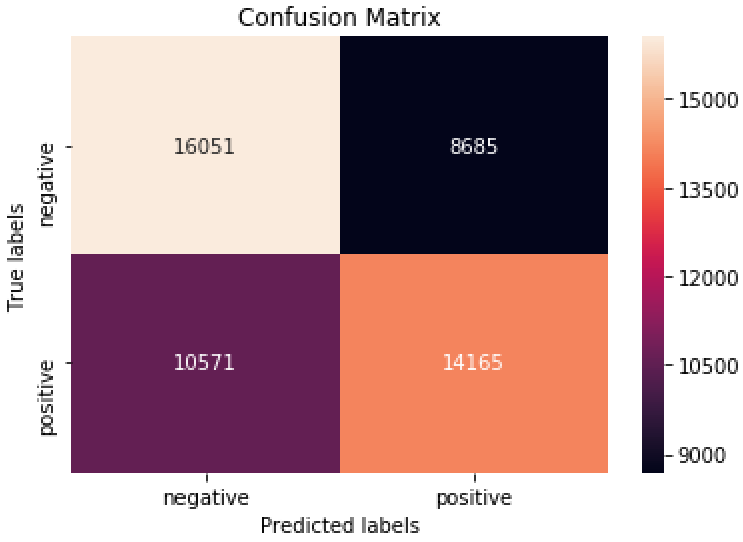
```
============================================================================
========================
Train confusion matrix
the maximum value of tpr*(1-fpr) 0.3715863303679612 for threshold 0.508
the maximum value of tpr*(1-fpr) 0.3715863303679612 for threshold 0.508
[[16051  8685]
 [10571 14165]]
Test confusion matrix
the maximum value of tpr*(1-fpr) 0.3678914811653749 for threshold 0.532
[[2384  881]
 [9643 8542]]
```

In [0]:

```
import seaborn as sn
import matplotlib.pyplot as plt
heat_map = plt.subplot()
sn.heatmap(conf_matrix, annot=True, ax = heat_map, fmt='g')

heat_map.set_ylabel('True labels')
heat_map.set_xlabel('Predicted labels')
heat_map.set_title('Confusion Matrix')
heat_map.xaxis.set_ticklabels(['negative', 'positive'])
heat_map.yaxis.set_ticklabels(['negative', 'positive'])
```

Out[138]:

```
[Text(0, 0.5, 'negative'), Text(0, 1.5, 'positive')]
```



3. Conclusion

In [0]:

```
from prettytable import PrettyTable

x = PrettyTable()

x.field_names = ["Vectorizer", "Hyperparameter(C)", "AUC"]

x.add_row(["BOW", 0.001, 0.7059])
x.add_row(["TFIDF", 0.1, 0.6989])
x.add_row(["Avg W2V", 1, 0.6671])
x.add_row(["TFIDF W2V", 0.01, 0.6757])
x.add_row(["One Hot", 0.01, 0.6313])

print(x)
```

```
+------------+-------------------+--------+
| Vectorizer | Hyperparameter(C) |  AUC   |
+------------+-------------------+--------+
|    BOW     |       0.001       | 0.7059 |
|   TFIDF    |        0.1        | 0.6989 |
|  Avg W2V   |         1         | 0.6671 |
| TFIDF W2V  |        0.01       | 0.6757 |
|  One Hot   |        0.01       | 0.6313 |
+------------+-------------------+--------+
```